UNIVERSIDAD AUTÓNOMA DE MADRID
ESCUELA POLITÉCNICA SUPERIOR

TRABAJO FIN DE MÁSTER

Support Vector Machines and Multi-Task Learning

Máster Universitario en Investigación e Innovación en TIC

Autor: Carlos Ruiz Pastor

Tutor: José R. Dorronsoro Ibero
Cotutor: Carlos M. Alaíz Gudín
Departamento de Ingeniería Informática

Madrid, September 4, 2018

Contents

List of Figures

List of Tables

List of Algorithms

Resumen

En la mayoría de casos los problemas que se resuelven utilizando aprendizaje automático no están aislados, sino que hay numerosas tareas similares con las que están relacionados. El aprendizaje multitarea es una aproximación del aprendizaje automático que trata de resolver múltiples tareas al mismo tiempo, logrando así una perspectiva más amplia del problema global. Las máquinas de vectores soporte son unos de los métodos más populares en aprendizaje automático, y han demostrado ser modelos útiles que además están apoyados por la teoría de optimización convexa.

En este trabajo estudiamos la adaptación de las máquinas de vectores soporte con el objetivo de encajar en un entorno multitarea. Nuestro primer paso para lograr esto es explorar la teoría de optimización convexa, específicamente las máquinas de vectores soporte. Hacemos un resumen de los problemas de optimización, incluyendo las condiciones KKT, así como el truco del kernel y el algoritmo SMO, que es el más popular para entrenar SVMs. Basamos nuestro estudio en dos adaptaciones previas de las SVMs al aprendizaje multitarea, desarrollando sus ideas y presentando las similitudes y diferencias entre ambas aproximaciones. También analizamos la relación entre los enfoques para obtener las ventajas que cada uno ofrece.

Para probar la precisión de las SVMs multitarea proponemos dos experimentos utilizando datos reales. El primero utiliza datos de estudiantes de institutos de Inglaterra y el objetivo es predecir los resultados de los estudiantes en un test específico. En estos experimento se utilizan datos de estudiantes de 139 institutos, y la predicción de las notas en cada instituto se observa como una tarea distinta. El segundo experimento consiste en predecir la producción solar, medida como un porcentaje, en dos islas de España: Mallorca y Tenerife. Aunque ambas islas pertenecen a España la distancia entre ellas es de más de 2.200 km. Esto hace que las predicciones en cada isla sean tareas separadas. En estos experimentos comparamos el enfoque multitarea con un enfoque global monotarea y, cuando es posible, entrenando un modelo distinto para cada tarea.

A partir de los resultados obtenidos podemos observar que el enfoque multitarea obtiene mejores resultados que el enfoque global monotarea. En el caso de los datos de institutos obtenemos un MAE de 8,226 puntos usando el enfoque monotarea y uno de 8,039 puntos con el multitarea. Además, esta mejora tiene lugar no solo globalmente sino también en la comparación tarea a tarea el enfoque multitarea obtiene mejores resultados en 90 de las 139 tareas. En el experimentos solar obtenemos una mejora de 0.45 % en Tenerife usando el modelo multitarea, mientras que en Mallorca el error es 0.15 % mayor; por tanto, se obtiene globalmente una mejora de 0.15 %. Estos resultados dan la motivación para una investigación futura con el objetivo de desarrollar todo el potencial de las SVMs multitarea.

Abstract

In most cases the problems we solve using machine learning are not isolated; instead there are several similar and related tasks. Multi-task learning is an approach of machine learning that tries to solve multiple related tasks at the same time, achieving a broader perspective of the global problem. Support vector machines are one of the most popular methods in machine learning, and they have proven to be really useful models which are also supported by the convex optimization theory.

In this work we study the adaptation of support vector machines in order to match a multi-task learning framework. Our first step to achieve this goal is to explore convex optimization theory and more specifically, support vector machines theory. We will give an overview of the optimization problems, including the KKT conditions, as well as the kernel trick and the SMO algorithm, the most used algorithm to train SVMs. We base our study in two previous adaptations of SVMs to multi-task learning, developing their ideas and presenting the similarities and differences between the two approaches. We also analyze the relation between the approaches and the possible advantages each one offers.

To test the accuracy of multi-task SVMs we propose two experiments using real data. The first one uses data from English school students and the goal is to predict the results of the students in a specific test. In this experiment, 139 different schools are used and predicting the marks of their students in each one of them is seen as a different task. Our second experiment consists on predicting the solar production measured as a percentage in two different islands of Spain: Majorca and Tenerife. Although both islands are part of Spain, the distance between both is roughly 2,200 km, making the predictions separate tasks. In our experiments we compare the multi-task approach with a single-task global approach and, when possible, with a model specialized for each task.

From the results obtained we can observe that multi-task approach gets better results than a single-task global approach. In the case of the school data we get a MAE of 8.226 points using the single-task approach while the multi-task approach achieves 8.039. Moreover, we see that this improvement takes place not only globally, but in a task by task comparison the multi-task approach performs better in 90 tasks out of 139. In the solar experiment we obtain an improvement of 0.45% in the MAE of the prediction in Tenerife using the multi-task model, while the error in Majorca is 0.15% worse than using a single-task approach, resulting in a global improvement of 0.15%. This results give the motivation to do further research in this area with the goal of finding the full potential of multi-task SVMs.

Acknowledgements

En primer lugar, quiero agradecer a mi tutor, José R. Dorronsoro, su dedicación y paciencia porque, a pesar de los abundantes compromisos, siempre es capaz de sacar un rato para ayudar o corregir los fallos. También agradezco su supervisión y ayuda durante todo el año en el que me ha guiado y cuyo resultado es este trabajo, en el que siempre ha puesto en primer lugar mi aprendizaje y formación. Aprecio que, a pesar de su exigencia, ha valorado mi trabajo y me ha dado ánimos para continuar por esta senda.

Agradezco también a mi cotutor Carlos M. Aláiz del que destaco su atención en los detalles y sus geniales ideas. Ha sido de gran ayuda tanto mostrándome los fallos que tengo como ofreciendo soluciones. También aprecio su capacidad para entenderme rápidamente y sus acertadas aportaciones, haciendo que sea realmente fácil trabajar con él.

Agradezco a Sara Dorado su contante apoyo, aprecio el hecho de haberme escuchado cuando tenía dudas y estar siempre para ayudarme. Doy también las gracias a mi compañero Alejandro Catalina, del que destaco su motivación e interés, ya que siempre ofrece una mano de forma altruista.

Doy las gracias a la Cátedra UAM-IIC de Ciencia de Datos y Aprendizaje Automático por la ayuda de máster concedida. Por último, quiero agradecer a mi familia su apoyo que me permite dedicarme a la investigación sin pedirme nada a cambio.

Chapter 1

Introduction

In this chapter we will first present the motivation behind this work, which is mainly to study the adaptation of the SVM's theory to a multi-task learning framework. We will also define the objectives we aim to achieve to do this. Finally, we will describe the structure followed in this work.

1.1 Motivation

Multi-task learning is a field inside the transfer learning theory, where the goal is to combine knowledge from different models in order to have a larger quantity of information to learn from. In particular, multi-task learning aims to combine different datasets from problems that are related, which receive the name of tasks; then, all tasks are solved at the same time and the solution of each single one makes use of the knowledge of the remaining tasks. That way, the amount of information that we can use is not constrained to the data we have collected for any specific problem; instead, we can take advantage of data collected for similar problems. This is especially useful in those cases where the amount of data avalaible is not enough to train a single operative model for each task, but the combination of multiple datasets is sufficient. On the other side, Support Vector Machines (SVMs) are popular models in machine learning due to its multiple characteristics and the theory that supports them. Our main motivation for this work is to study the adaptation SVMs in order to use them in a multi-task learning framework.

1.2 Objectives

The objective of this work is two-fold: in first place, based on previous works [1, 2], we want to connect the different approaches that have been made and to formalize the multi-task SVM. In the second place, we put our focus on the comparison of the multi-task SVR and traditional single-task SVRs. For our first goal, we will study the ideas presented in [1] and the posterior approach of [2] and we show its similarities and differences. Although at first glance both works are not connected we will see that the second one can be formulated as a generalization of the first one. For the second goal, we will carry out experiments using two different datasets. In the first place, we will work with a dataset containing data from high schools of England that has been traditionally used for multi-task purposes [3, 1, 4]. In the second place, we will use a multi-task dataset, in which we combine the solar production of the islands of Majorca and Tenerife in Spain, interpreting each island's solar production forecast as a separate task. In order to carry out these experiments the implementation of an SVR of `Scikit-learn` will be used. However, in order to make an easy implementation

of the multi-task SVR a hybrid approach between those of [1] and [2] will be developed and used.

1.3 Structure

This work is structured as follows: in Chapter 2 we will cover the mathematical fundamentals of SVR, mainly, the kernel induced feature spaces in Section 2.1 and the optimization theory in Section 2.2; in Section 2.3 we will also develop the theory of the traditional single-task SVM; finally we will present the SMO algorithm in Section 2.4, an algorithm used to train SVMs. In Chapter 3 we will develop the theory of the multi-task SVRs. In the first place, in Section 3.1 we will present the approach made in [1], which is called *Regularized Multi-Task Learning*, where the kernel used is linear and the bias is omitted. In Section 3.2 we will develop the ideas of [2] and we will show the connection with *Regularized Multi-Task Learning* as well. We will also present the algorithm needed to train the multi-task SVR that receives the name of Generalized SMO. In Chapter 4 we present the experiments carried out and their results. First, in Section 4.1 we explain the implementation used for the experiments; after this, Section 4.2 contains the results of the experiment that use the school dataset, whereas in Section 4.3 we present the experiment carried out using the solar production datasets. Finally, in Chapter 5 we will discuss the results obtained and we will present the ideas that are left for further work.

Chapter 2

Support Vector Machines

This chapter has the purpose of covering the theory necessary for the rest of this work. In first place it gives an overview of the kernel-induced feature spaces and optimization theory in Section 2.1 and Section 2.2 respectively; both topics are very important because they lie underneath the theory of Support Vector Machines. Support Vector Machines (SVMs) are explained in the third section of this chapter as they are presented as an application of the general convex theory covered previously. We will also see the motivation that makes SVMs interesting models in a machine learning framework, as well as the algorithm SMO, which is the most popular one for training these models. It is important to remark that this chapter has its focus on single-task classical SVMs and does not cover multi-task SVMs, which will be explained in Chapter 3.

2.1 Kernel-Induced Feature Spaces

In machine learning, specifically in supervised classification or regression, each pattern x, a point of a feature space $\mathcal{X}$ which is usually a subspace of $\mathbb{R}^d$, is assigned a label or target; our goal is to learn the underlying relations in order to assign labels or targets to new examples. To do this several hypotheses can be chosen, and among all, linear hypotheses are the simplest and the easiest to explain. Linear models have multiple advantages such as explainability and often explicit solutions of the problem. Because of this they are one of the most popular examples of machine learning algorithms and historically the first to appear; see for example linear regression or Rosenblatt's perceptron [5] algorithms, which are linear algorithms for regression and classification respectively. Nevertheless, linear models are very limited and usually they are not expressive enough for real-world applications. Real-world data include complex relations that may not be linear and thus it is necessary a model that is capable of detecting those non-linear relations. One possible solution for introducing non-linearity to models is to map the original points on $\mathcal{X}$ to a large dimensional space $\mathcal{F}$ through some mapping function ϕ, that is

$$\begin{aligned} \phi\colon \mathcal{X} \subset \mathbb{R}^d &\to \mathcal{F} \subset \mathbb{R}^D \\ x = (x^1, \dots, x^d)^t &\mapsto \phi(x) = (\phi(x)^1, \dots, \phi(x)^D)^t \ , \end{aligned}$$

where D can be even infinite resulting then in an infinite dimensional space $\mathcal{F}$. At this point it is relevant to recall one important property that is usually present in linear models, that is the possibility of writing the models in terms of the Gramm matrix G. The matrix G is constructed using the inner products of the patterns. Given a data matrix X where each row is the transpose of a pattern $x_i \in \mathcal{X}$ of our sample of size N, then the Gramm

matrix is

$$G = \begin{pmatrix} \langle x_1, x_1 \rangle & \langle x_1, x_2 \rangle & \dots & \langle x_1, x_N \rangle \\ \langle x_2, x_1 \rangle & \langle x_2, x_2 \rangle & \dots & \langle x_2, x_N \rangle \\ \vdots & \vdots & \ddots & \vdots \\ \langle x_N, x_1 \rangle & \langle x_N, x_2 \rangle & \dots & \langle x_N, x_N \rangle \end{pmatrix} .$$

This means that we expect that the model is no longer dependent on the data itself but on the inner products of the patterns. This property paves the way for the introduction of kernel functions and kernel-induced features.

The problem of mapping the data on a larger space is its computational cost, that is linearly dependent on D, the dimension of the new space $\mathcal{F}$. This is where kernel functions play an important role.

Definition 2.1.1. *A kernel function k is a function such that for all points x, z in our original feature space $\mathcal{X}$ verifies*

$$k(x, z) = \langle \phi(x), \phi(z) \rangle ,$$

where ϕ is a mapping from the original space to a new feature space $\mathcal{F}$ induced by k.

If for a mapping function ϕ we know its corresponding kernel function k, then we have a direct way of computing the Gramm matrix on the new space $\mathcal{F}$ and moreover, it is no longer dependent on ϕ. For example, in the original space the kernel is

$$k(x, z) = \langle x, z \rangle ,$$

this is called a linear kernel, and $\phi(x) = x$. Another example is the polynomial kernel:

$$\begin{aligned} k(x, z) = (\langle x, z \rangle + c)^2 &= \left(\sum_{i=1}^{d} x^i z^i + c \right) \left(\sum_{j=1}^{d} x^j z^j + c \right) \\ &= \sum_{i=1}^{d} \sum_{j=1}^{d} x^i x^j z^i z^j + 2c \sum_{i=1}^{d} x^i z^i + c^2 . \end{aligned}$$

From this definition it is easy to see that the transformation whose inner product is this polynomial kernel is the following

$$\phi(x) = \left(x^1 x^1, x^1 x^2, \dots, x^1 x^d, \dots, x^d x^1, \dots, x^d x^d, \sqrt{2c} x^1, \dots, \sqrt{2c} x^d, c \right) ,$$

where the dimension D is $d^2 + d + 1$.

Although the use of a kernel function k is attractive, it seems that using this approach requires finding a complicated feature space and then work out the inner product until finding the function k in terms of the original features, which is not practical and definitely not trivial. However, in practice the approach is taken reversely: the kernel function is defined directly and hence the feature space is implicitly defined, which is much easier to work with. Nevertheless, to do this, it is necessary to know what are the properties that ensure that $k(x, z)$ is a kernel function for some appropiate feature space. The characterization of a kernel function is given by Mercer's Theorem [6].

Theorem 2.1.1 (Mercer's Theorem)**.** *Let $\mathcal{X} \subset \mathbb{R}^d$ and suppose that $k(x, z) : \mathbb{R} \times \mathbb{R} \mapsto \mathbb{R}$ is a continuous symmetric function such that the integral operator $T_k \colon L_2(\mathcal{X}) \to L_2(\mathcal{X})$ associated to k and defined as*

$$f(x) \mapsto (T_k f)(x) = \int_{\mathcal{X}} k(x, z) f(z) dz$$

is non-negative, which means

$$\int_{\mathcal{X}\times\mathcal{X}} k(x,z)f(x)f(z)dxdz \geq 0 \ , \forall f \in L_2(\mathcal{X}) \ . \tag{2.1.1}$$

Then we can expand $k(x,z)$ in a uniformly convergent series (on $\mathcal{X}\times\mathcal{X}$) in terms of T_k's orthonormal eigen-functions $\phi_j \in L_2(\mathcal{X})$ and non-negative associated eigenvalues $\lambda_j \geq 0$ as

$$k(x,z) = \sum_{l=1}^{\infty} \lambda_l \phi_l(x)\phi_l(z) \ .$$

The complete proof of Mercer's Theorem is out of the scope of this work. However, we can have an intuition. Note in first place that the operator T_k is self-adjoint, we can see it in the following way:

$$\begin{aligned}
\langle T_k f, g\rangle &= \int_{\mathcal{X}} (T_k f)(x)g(x)dx = \int_{\mathcal{X}} \left(\int_{\mathcal{X}} k(x,z)f(z)dz \right) g(x)dx \\
&= \int_{\mathcal{X}} \left(\int_{\mathcal{X}} k(z,x)f(z)dz \right) g(x)dx = \int_{\mathcal{X}} \int_{\mathcal{X}} k(z,x)g(x)f(z)dzdx \\
&= \int_{\mathcal{X}} f(z) \left(\int_{\mathcal{X}} k(z,x)g(x)dx \right) dz = \langle f, T_k g\rangle \ .
\end{aligned}$$

Moreover, it can be proven that T_k is a compact operator; hence; we can apply the Spectral Theorem for compact operators on Hilbert Spaces [7, Volume 1, Chapter 8]:

Theorem 2.1.2 (Spectral Theorem)**.** *Suppose T is a compact self-adjoint operator on a Hilbert space $\mathcal{F}$. Then there is an orthonormal basis of $\mathcal{F}$ consisting of eigenvectors $\phi_i(x) \in L_2(\mathcal{X})$ of T real eigenvalues λ_i such that*

$$\lambda_i \phi_i(x) = (T\phi_i)(x) \ .$$

Furthermore, it can be seen that

$$\sum_{i=1}^{\infty} \lambda_i \left\| \phi_i(x)\phi_i(z) \right\| < \infty \ ,$$

and therefore,

$$\sum_{i=1}^{\infty} \lambda_i \phi_i(x)\phi_i(z)$$

is uniformly convergent and it can be proved that it converges to $k(x,z)$. With the result of Mercer's Theorem, the feature space $\mathcal{F}$ is $\text{span}\,\{\phi(x) : x \in \mathcal{X}\}$ and the feature map ϕ from the original space $\mathcal{X}$ to the new feature space $\mathcal{F}$ is automatically given by

$$\phi(x) = \left(\sqrt{\lambda_1}\phi_1(x), \sqrt{\lambda_2}\phi_2(x), \ldots, \sqrt{\lambda_k}\phi_k(x), \ldots \right) \ .$$

Moreover, a more practical characterization of a kernel can be given with the following definition of kernel function also given by Mercer.

Definition 2.1.2 (Kernel Function Characterization)**.** *Given a non-empty space $\mathcal{X}$ we say that a symmetric function $k(x,y) : \mathcal{X}\times\mathcal{X} \mapsto \mathbb{R}$ is a kernel if for any finite sample $x_1, \ldots, x_n \in \mathcal{X}$ the matrix*

$$K = (k(x_i, x_j))_{i,j} \ , i = 1, \ldots, n, \ j = 1, \ldots, n$$

is positive semi-definite.

2.2 Optimization Theory

2.2.1 Problem Formulation

Most problems in machine learning aim to find an element that maximizes or minimizes some functional. In the case of linear models this is finding a vector w that minimizes an objective function, often restricted to some constraints. The optimization theory has the goal of characterizing such solutions, and moreover, to develop algorithms to find them efficiently. An important example of this is the duality theory developed for linear models, which gives easy-to-check characterization rules for a solution of the problem. It is particularly interesting the case of convex optimization due to the inherent properties of convexity. The general problem is to find the solution w^* in a set Ω that minimizes some objective function given some constraints.

Definition 2.2.1 (Primal Problem)**.** *Given the functions f, g_i, h_j, $i \in \{1, \dots, k\}$, $j \in \{1, \dots, m\}$ defined in $\Omega \in \mathbb{R}^d \mapsto \mathbb{R}$, the primal optimization problem is defined as*

$$\begin{aligned} &\min_{w \in \Omega} f(w) \\ &s.t. \quad g_i(w) \leq 0, i \in \{1, \dots, k\} \ , \\ &\qquad h_j(w) = 0, j \in \{1, \dots, m\} \ . \end{aligned}$$

where $f(w)$ is the objective function and g_i, h_i are the inequality and equality constraints, respectively.

For the sake of simplicity we will use the following notation

- $g(w) \leq 0$ instead of $g_i(w) \leq 0, i \in \{1, \dots, k\}$.
- $h(w) = 0$ instead of $h_j(w) = 0, j \in \{1, \dots, m\}$.

The region of the space where all the constraints are satisfied is called the feasible region.

Definition 2.2.2 (Feasible Region)**.** *The feasible region is the region of the domain of $f(w)$ where all the restrictions are fulfilled. That is,*

$$\mathcal{R} = \{w \in \Omega : g(w) \leq 0, \ h(w) = 0\} \ .$$

Given the definition of feasible region, we can state the definition of global and local solutions.

Definition 2.2.3 (Global and local solution)**.** *We say that $w^* \in \mathcal{R}$ is a **global solution** of the optimization problem if $f(w^*) \leq f(w) \ \forall w \in \mathcal{R}$. Moreover, we can say that w^* is a **local solution** if $\exists \epsilon$ such that $f(w^*) \leq f(w) \ \forall w \in \mathcal{R}$, $\|w - w^*\| < \epsilon$.*

Any solution w^* must fulfill the equality and inequality conditions; an inequality condition $g_i(w)$ is said to be **active** if $g(w^*) = 0$ and **inactive** if $g(w^*) < 0$. While equality constraints are always active, this is not the case for inequality constraints. In order to transform a constraint from an inequality to an equality, slack variables ξ_i are added. In that way, the restriction $g_i(w) \leq 0$ is transformed to $g_i(w) + \xi_i = 0, \ \xi_i \geq 0$. These slack variables indicate the looseness of the constraint g_i in a particular solution; therefore, an active constraint will have $\xi_i = 0$. Moreover, an interesting result that we will prove below is that a convex function ensures that any local minimum is a global one; therefore simplifying the task of finding the global minimum. This property, although very likeable, is not fulfilled in general, which is the reason why the optimization theory is closely related to convexity; thus it is important to recall some definitions concerning convexity.

Definition 2.2.4 (Convex Set)**.** *A set Ω is said to be convex if and only if $\forall u, v \in \Omega$ and $\lambda \in (0,1)$*

$$\lambda u + (1-\lambda)v \in \Omega \, . \tag{2.2.1}$$

Notice that the intersection of two convex sets is also convex. Given this definition is immediate to define a convex function.

Definition 2.2.5 (Convex Function)**.** *A function $f : \mathcal{X} \subset \mathbb{R}^d \mapsto \mathbb{R}$ is convex if and only if its epigraph, that is*

$$\left\{(x,t) \in \mathbb{R}^{d+1}, x \in \mathbb{R}^d \, , t \geq f(x)\right\} \, ,$$

is a convex set. Equivalently, f is convex if and only if $\forall x, y \in \mathcal{X}$ and $\lambda \in (0,1)$

$$f\left(\lambda x + (1-\lambda)y\right) \leq \lambda f(x) + (1-\lambda) f(y) \, . \tag{2.2.2}$$

Now, it is easy to see that local minimum of a convex function implies global minimum.

Proposition 2.2.1. *Any local minimum w^* of a convex function $f(w)$ is also a global minimum of $f(w)$.*

Proof. Let $w^* \in \Omega$ be a local minimum, then $f(w^*) \leq f(\lambda u + (1-\lambda)w^*)$ $\forall u \in \Omega$ for a small enough $\lambda > 0$. Since f is convex we can write the following

$$f(w^*) \leq f(\lambda u + (1-\lambda)w^*) \leq \lambda f(u) + (1-\lambda) f(w^*) \, ,$$

which implies

$$f(w^*) \leq f(u) \, , \forall u \in \Omega \, .$$

□

Any optimization problem in which the set Ω, the objective function f and all the constraints are convex is said to be a *convex problem*. It is easy to see that the feasible region of a convex problem is convex since it is the intersection of the set Ω with the epigraphs of the constraints functions, which are convex. Since our goal is to work with SVMs, we will restrict ourselves to a simpler case where all the restrictions are linear.

2.2.2 Lagrangian Theory

The Lagrangian theory, developed firstly by Fermat and then generalized by Lagrange, aims to characterize the solutions of the optimization problems. In order to get a better understanding we will start with the simplest case, where there are no constraints. In this case the Fermat Theorem [8] applies.

Theorem 2.2.1 (Fermat's Theorem)**.** *A necessary condition for w^* to minimize $f(w) \in C^1$, is $\nabla_w f(w^*) = 0$. If we assume $f(w)$ a convex function, then $\nabla_w f(w^*) = 0$ is a necessary and sufficient condition for w^* to minimize f.*

It is well known that a point w^* where $\nabla_w f = 0$ is a critical point, and, if the function is convex it must be a global minimum. However, when some restrictions are imposed, the condition $\nabla_w f = 0$ might not be achieved in the feasible region. To solve this we can notice some aspects of the problem that can help us to have a better intuition. We want to advance in the opposite direction of the one indicated by the gradient $\nabla_w f$. However, since the constraints have to be met, this may not be done freely. Let us restrict ourselves first to equality constraints $h_j(w) = 0$, which can be seen as the zero-level curve of $h_j(w)$;

therefore, $h_j(w) = 0$ is a curve such that its tangent vector is perpendicular to $\nabla h_j(w)$ at every point in it. If a point is in the feasible region and we want to move to another point, we have to do it perpendicularly to $\nabla h_j(w)$. This means that, intuitively, in order to minimize $f(w)$ we have to move in the component of the gradient $\nabla_w f$ that is perpendicular to ∇h_j. When there are multiple constraints we have to move perpendicularly to

$$\mathcal{H} = \text{span}\left\{\nabla h_j(w)\ , j \in \{1, \ldots, m\}\right\} .$$

This can be done only if $\nabla_w f(w) \notin \mathcal{H}$; hence, we can express this condition in a simple way: w^* is a solution of the problem with constraints if $\exists \beta_1, \ldots, \beta_m$ such that

$$\nabla_w f(w^*) = \sum_{j=1}^{m} \beta_j \nabla h_j(w^*) \ ; \tag{2.2.3}$$

thus, given a problem with objective function f and equality constraints h_j we can define its *Lagrangian* as

$$\mathcal{L}(w, \beta) = f(w) + \sum_{j=1}^{m} \beta_j h_j(w) \ .$$

Given this definition and the previous results it is easy to prove the following theorem [8, Chapter 5, Theorem 5.8].

Theorem 2.2.2 (Lagrange)**.** *Necessary conditions of w^* for being a solution of a problem with objective function $f(w)$ and equality constraints $h_j(w) = 0,\ j \in \{1, \ldots, m\}$ are*

$$\nabla_w \mathcal{L}(w^*, \beta^*) = 0 \ ,$$
$$\nabla_\beta \mathcal{L}(w^*, \beta^*) = 0 \ ,$$

for some values β^*. These conditions are sufficient if $f(w)$ and $h_j(w),\ j \in \{1, \ldots, m\}$, are convex.

It is clear to see that the first condition is the condition (2.2.3) while the second one ensures the feasibility of w^*.

We now consider the more general case of an optimization problem, with both equality and inequality constraints. Then, we can give the following definition.

Definition 2.2.6 (Lagrangian)**.** *Given an optimization problem with objective function $f(w)$, equality constraints $h_j(w) = 0,\ j \in \{1, \ldots, m\}$ and inequality constraints $g_i(w) \leq 0\ , i \in \{1, \ldots, k\}$ we define the Lagrangian of such problem as:*

$$\mathcal{L}(w, \beta) = f(w) + \sum_{i=1}^{k} \alpha_i g_i(w) + \sum_{j=1}^{m} \beta_j h_j(w) = f(w) + \alpha g(w) + \beta h(w) \ ,$$

where $\alpha \geq 0$ and β are called the Lagrange multipliers.

From the definition of Lagrangian we can define the Dual Function.

Definition 2.2.7 (Dual Function)**.** *Given an optimization problem and its Lagrangian $\mathcal{L}(w, \alpha, \beta)$ we define its Dual Function as*

$$\Theta(\alpha, \beta) = \inf_{w \in \Omega} \mathcal{L}(w, \alpha, \beta) \ .$$

Now we can define the Dual Problem.

Definition 2.2.8 (Dual Problem)**.** *The Dual Problem of a Lagrangian $\mathcal{L}(w, \alpha, \beta)$ is defined as:*

$$\max_{\alpha,\beta} \Theta(\alpha, \beta)$$
$$s.t.\ \alpha \geq 0\ ,$$

where $\Theta(\alpha, \beta)$ is the Dual Function.

To connect the primal and dual problem we have the duality theorems. The first one, Weak Duality Theorem [8, Chapter 5, Theorem 5.15], is the following.

Theorem 2.2.3 (Weak Duality Theorem)**.** *Let (α, β) and w be feasible solutions (not necessarily optimal) for the dual and primal problems respectively; then*

$$\Theta(\alpha, \beta) \leq f(w)\ .$$

Proof. Using the definition of the dual function and the feasibility of w and α,

$$\Theta(\alpha, \beta) = \inf_{w\in\Omega} \mathcal{L}(w, \alpha, \beta) \leq f(w) + \sum_{i=1}^{k} \alpha_i g_i(w) + \sum_{j=1}^{m} \beta_j h_j(w) \leq f(w)\ .$$

□

This theorem provides two useful corollaries.

Corollary 2.2.1. *The optimal value $\Theta(\alpha^*, \beta^*)$ of the dual is upper bounded by the optimal value of the primal $f(w^*)$, that is*

$$\Theta(\alpha^*, \beta^*) \leq f(w^*)\ .$$

Corollary 2.2.2. *Any tuple (α^*, β^*) and w^* such that*

$$\Theta(\alpha^*, \beta^*) = f(w^*)$$

are optimal solutions for both the dual and the primal problems respectively.

The difference between the optimal values of the primal and the dual is called the *dual gap* and therefore, having a null dual gap is a sufficient condition (although not necessary in general) to find optimal solutions. This idea motivates the second duality theorem [8, Chapter 5, Theorem 5.20].

Theorem 2.2.4 (Strong Duality Theorem)**.** *Given a convex problem*

$$\begin{aligned} &\arg\min_{w\in\Omega} f(w) \\ s.t.\quad & g_i(w) \leq 0, i \in \{1, \ldots, k\}\ , \\ & h_j(w) = 0, j \in \{1, \ldots, m\}\ , \end{aligned}$$

if g_i and h_j are affine functions, then the dual gap of the problem is zero.

This theorem has the following consequence: given a solution (α^*, β^*) of the dual problem, we know the minimum value of the primal function $f(w^*) = \Theta(\alpha^*, \beta^*)$. Moreover, in some particular cases where we have an equation that expresses w in terms of (α, β), we can recover the primal solution w^* from (α^*, β^*). After this result, we can present the Karesh-Kuhn-Tucker conditions [9] to characterize the optimal solution of an optimization problem.

Theorem 2.2.5 (KKT Theorem)**.** *Suppose a convex optimization problem with affine constraints. Then, w^* is an optimal solution of the primal problem if and only if there exist (α^*, β^*) such that:*

$$\begin{aligned}
\frac{\partial \mathcal{L}(w, \alpha, \beta)}{\partial w}\Big|_{w^*, \alpha^*, \beta^*} &= 0 \ , \\
\frac{\partial \mathcal{L}(w, \alpha, \beta)}{\partial \beta}\Big|_{w^*, \alpha^*, \beta^*} &= 0 \ , \\
\alpha_i^* g_i(w^*) &= 0, \ i \in \{1, \dots, k\} \ , \\
g_i(w^*) &\leq 0, \ i \in \{1, \dots, k\} \ , \\
\alpha_i^* &\geq 0, \ i \in \{1, \dots, k\} \ .
\end{aligned}$$

Proof. Since the KKT conditions are necessary and sufficient we will split the proof for both cases. We will first check necessity; if w^* is an optimal solution of the primal problem, by the Strong Duality Theorem 2.2.4, we know that

$$\Theta(\alpha^*, \beta^*) = \mathcal{L}(w^*, \alpha^*, \beta^*) = f(w^*) \ ,$$

and since

$$\Theta(\alpha^*, \beta^*) = \inf_{w \in \Omega} \mathcal{L}(w, \alpha^*, \beta^*) \ ,$$

w^* must be a minimum of $\mathcal{L}$, which is convex on w; therefore, we have

$$\frac{\partial \mathcal{L}(w, \alpha^*, \beta^*)}{\partial w}\Big|_{w^*} = 0$$

(first KKT condition). Since w^* is a solution, it is feasible; therefore

$$\frac{\partial \mathcal{L}(w^*, \alpha^*, \beta)}{\partial \beta}\Big|_{\beta^*} = h(w^*) = 0$$

(second KKT condition), and $g_i(w^*) \leq 0, \ i \in \{1, \dots, k\}$ (fourth KKT condition). Finally, using the Weak Duality Theorem we can write the following,

$$\Theta(\alpha, \beta) \leq \mathcal{L}(w^*, \alpha, \beta) \leq f(w^*) \ ,$$

for every α, β. However, using the Strong Duality Theorem, it must exist a solution (α^*, β^*) with $\alpha^* \geq 0$ (last KKT condition) of the dual that fulfills

$$\Theta(\alpha^*, \beta^*) = \mathcal{L}(w^*, \alpha^*, \beta^*) = f(w^*) \ ;$$

then, using the second equality we have

$$f(w^*) + \sum_{i=1}^{k} \alpha_i^* g_i(w^*) + \sum_{j=1}^{m} \beta_j^* h_j(w^*) = f(w^*) \ .$$

Using $g_i(w^*) \leq 0$ and $\alpha > 0$ we have then

$$\alpha_i g_i = 0 \ , i \in \{1, \dots, k\} \ ,$$

which is the third KKT condition.

To prove sufficiency, we need to see that w^* is an optimal solution of the primal problem. It is therefore sufficient to see $\Theta(\alpha^*, \beta^*) = f(w^*)$. Using the first condition we have

$$\Theta(\alpha^*, \beta^*) = \inf_{w \in \Omega} \mathcal{L}(w, \alpha^*, \beta^*) = \mathcal{L}(w^*, \alpha^*, \beta^*)$$

Using the remaining properties we can write the following

$$\mathcal{L}(w^*, \alpha^*, \beta^*) = f(w^*) + \sum_{i=1}^{k} \alpha_i^* g_i(w^*) + \sum_{j=1}^{m} \beta_j^* h_j(w^*) = f(w^*) \ ,$$

where the first equality is just the definition and the second one uses the second and third KKT conditions. □

2.3 Support Vector Machines

We will dedicate this section to the motivation and analysis of Support Vector Machines, which are one of the most popular models in machine learning. Their popularity can be related to their idea of maximum separability and the extensive theory that supports this. The first subsection aims to present the idea that motivates SVMs and their utility. In the second subsection we will perform a more detailed analysis of the properties of SVMs. Finally, the last subsection is devoted to the SMO algorithm, which is currently the most popular method for solving SVMs.

2.3.1 Motivation

Before showing the SVMs motivation, it is necessary to define some concepts. In first place, a **binary classification problem** is a problem in which we have patterns $x_i \in \mathcal{X} \subset \mathbb{R}^d \ , i = 1, \dots, N$ labeled with $y_i \in \mathcal{Y} \ , i = 1, \dots, N$, where $|\mathcal{Y}| = 2$, for example $\mathcal{Y} = \{-1, 1\}$. The labels of the patterns are also named classes. Then, our goal is to find a rule r such that $r(x_i) = y_i$, or at least one that minimizes $|\{i : r(x_i) \neq y_i\}|$. A binary classification problem is said to be **separable** if we can find such rule in a way that $r(x_i) = y_i$ for $i = 1, \dots, N$. Moreover, a problem is said to be **linearly separable** if it is separable and the rule r is linear on x; that is, it only uses linear combinations of the components of x. The linear rule that is typically used is $r(x) = \text{sign}\,(wx + b)$, where w is a vector of $\mathbb{R}^d$. The equation $wx + b = 0$ defines a plane in $\mathbb{R}^d$; thus, the rule $r(x) = \text{sign}\,(wx + b)$ results in a plane that divides the space in two halves, one for each class.

Support Vector Machines arise from the goal of, given a binary classification problem that is linearly separable, finding the optimal separating plane. The optimality property of a separating plane has to do with distance to the points of the problem. More precisely, a separating plane is considered optimal when it maximizes the distance to its closest points among the points of the classification problem. The idea behind this is that, if the sample is slightly changed it will be still correctly classified at the right side of the separating plane because there is some 'margin'. We can formalize this idea as follows: let assume that we have the following linearly separable sample,

$$\{(x_1, y_1), \dots, (x_N, y_N)\} \ , x_i \in \mathcal{X} \subset \mathbb{R}^d \ , y \in \{-1, 1\} \ ,$$

where x_i are the patterns and y_i their classes or labels. Then, the **margin** m of a separating plane $wx + b = 0$ is the minimum distance among the points of the sample; that is,

$$m = \min_{x_i} \ \frac{1}{\|w\|} \left| wx_i + b \right| \ ,$$

as it can be observed in Figure 2.3.1. However, the sample points are constrained, given that $wx + b$ is a separating plane, to

$$y_i(wx_i + b) \geq 0 \ , i = 1, \dots, N \ .$$

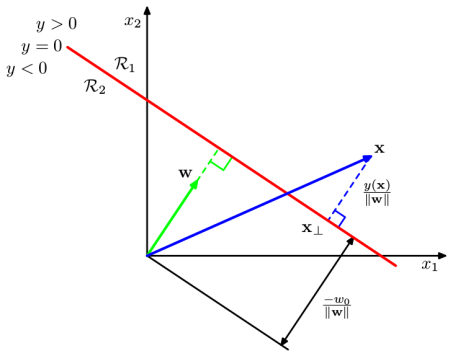

Figure 2.3.1: Image from Bishop's book [10, Chapter 7.1].: The plane $y(x) = wx + b = 0$ is shown in red. The vector w is shown in green and the vector x in blue. It illustrates the separation of the space in two halves and the distances from one point to the plane $y(x) = 0$.

With these definitions, if we want to look for the optimal plane of a linearly separable problem, we can write it as follows:

$$\begin{aligned} &\arg\max_{w\in\mathbb{R}^d} m \\ &\text{s.t.} \qquad \frac{1}{\|w\|} y_i(wx_i + b) \geq m,\ i = 1, \dots, N\ , \end{aligned}$$

which, multiplying by $\|w\|$ in the restrictions, is equivalent to

$$\begin{aligned} &\arg\max_{w\in\mathbb{R}^d} m \\ &\text{s.t.} \qquad y_i(wx_i + b) \geq \|w\|\, m,\ i = 1, \dots, N\ . \end{aligned}$$

Since the hyperplane is not dependent on the norm of w, we can set $\|w\| = 1/m$, obtaining the following,

$$\begin{aligned} &\arg\max_{w\in\mathbb{R}^d} \frac{1}{\|w\|} \\ &\text{s.t.} \qquad y_i(wx_i + b) \geq 1,\ i = 1, \dots, N\ . \end{aligned}$$

In order to take derivatives, it is useful to write the following equivalent problem,

$$\begin{aligned} &\arg\min_{w,b} \quad J(w, b, \xi) = \frac{1}{2}||w||^2 \\ &\text{s.t.} \qquad y_i(w \cdot x_i + b) \geq 1,\ i = 1, \dots, N\ . \end{aligned} \tag{2.3.1}$$

This way, we have written the problem of finding the optimal plane as an optimization problem. However, the assumption of linear separability may be too strong. When working with real world data, it is not common to find a sample that is linearly separable. To overcome this problem the **slack variables** ξ are introduced. These slack variables allow the points of the sample to be inside the margin or even misclassified. The problem with slack variables is the following:

$$\begin{aligned} &\arg\min_{w,b,\xi} \quad J(w, b, \xi) = C\sum_{n=1}^{N} \xi_i + \frac{1}{2}||w||^2 \\ &\text{s.t.} \qquad y_i(w \cdot x_i + b) \geq 1 - \xi_i,\ i = 1, \dots, N, \\ &\qquad\qquad \xi_i \geq 0,\ i = 1, \dots, N\ . \end{aligned} \tag{2.3.2}$$

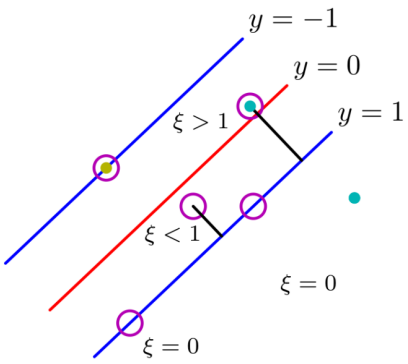

Figure 2.3.2: Image from Bishop's book [10, Chapter 7.1]: There are points of two classes, blue is 1 and yellow is -1. The separating plane is the line in red and the two lines in blue represent the margin with respect to the separating plane. The picture shows the regions of the space where a slack variable for a blue point takes different values.

With this definition, we can see that a point x_i that is correctly classified and further than the margin to the separating plane will have a slack variable $\xi_i = 0$. However, if the point is not far enough from the separating plane or it is misclassified it will have a slack variable of value $\xi_i \leq 1$ or $\xi_i > 1$ respectively, as shown in Figure 2.3.2. In the problem (2.3.2) there are two terms in the objective function. The first one is the sum of the slack variables, that is, the error made; while the second one, as we have defined it, is the inverse of the margin. That way, the parameter C has the goal of tuning the importance of the error over the margin. A large value of C penalizes the error and thus, it imposes a smaller margin. On the other hand, a small value of C allows a greater error and therefore, a greater margin. This formulation can also be seen from the perspective of regularization where the first term is the error and the second one is a regularization term of the model.

Until now we have worked with classifications problems. Nevertheless, this idea can also be extended to regression. In a regression problem, we have a sample with patterns x_i but instead of labels, each pattern has a target t_i. The goal is to find a regression function $r(x)$, that is a plane in a linear framework, that minimizes $\sum_{i=1}^{N} l(\|t_i - r(x_i)\|)$, where $l(z)$ is a non-negative loss function. Typically, the error function used is $e(z) = z^2$, this way, we try to minimize $\sum_{i=1}^{N} \|t_i - r(x_i)\|^2$, which has two important flaws: the first one is that we penalize every error, even if it is really small, the second one is that, being a quadratic function, the total sum can be affected by a single point that lies very far from the rest. The idea in SVMs for regression is to overcome these two problems by defining the following error function:

$$\hat{e}(z) = \begin{cases} -z - \epsilon & z < -\epsilon \\ 0 & -\epsilon \leq z \leq \epsilon \\ z - \epsilon & \epsilon < z \end{cases}, \tag{2.3.3}$$

for some $\epsilon > 0$. This function is shown in Figure 2.3.3. Looking at this error function we can remark two facts: the first one is that errors smaller than ϵ are not penalized, and the second is that it is piece-wise linear; thus, predictions far from the real value are not so determinant. Therefore, this error function achieves the desired properties we had previously asked. Given this, we can express the motivation behind the SVMs for regression. The idea is to find a regressor plane that has the points at a distance smaller

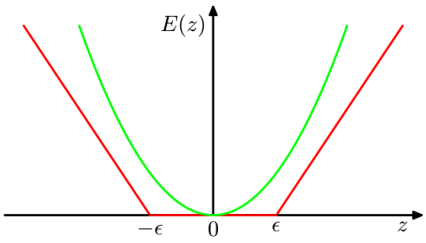

Figure 2.3.3: Image from Bishop's book [10, Chapter 7.1]: The quadratic error is shown in green and the function defined as $\hat{e}$ in red.

than ϵ, that is

$$w \cdot x_i + b \geq t_i - \epsilon \ ,$$
$$w \cdot x_i + b \leq t_i + \epsilon \ .$$

However, as in the previous case, we add some slack variables that allow the points to be outside the ϵ-tube around the regressor plane. This problem can be formulated also as an optimization problem:

$$\begin{aligned} \underset{w,b,\xi,\hat{\xi}}{\arg\min} \quad & J(w,\xi) = C\sum_{n=1}^{N}(\xi_i + \hat{\xi}_i) + \frac{1}{2}||w||^2 \\ \text{s.t.} \quad & w \cdot x_i + b \geq t_i - \epsilon - \hat{\xi}_i, \ i = 1,\dots,N \ , \\ & w \cdot x_i + b \leq t_i + \epsilon + \xi_i, \ i = 1,\dots,N \ , \\ & \xi_i, \hat{\xi}_i \geq 0, \ i = 1,\dots,N \ . \end{aligned}$$

It is important to note that we can define $\xi_i, \hat{\xi}_i$ as

$$\hat{\xi}_i = \begin{cases} t_i - \epsilon - w \cdot x_i - b & w \cdot x_i + b - t_i < -\epsilon \\ 0 & w \cdot x_i + b - t_i \geq -\epsilon \end{cases} \ ,$$
$$\xi_i = \begin{cases} 0 & w \cdot x_i + b - t_i \leq \epsilon \\ w \cdot x_i + b - t_i - \epsilon & \epsilon < w \cdot x_i + b - t_i \end{cases} \ .$$

Then we can observe two facts: in first place the slack variables $\xi, \hat{\xi}$ conform the error function defined in (2.3.3), that is

$$\xi_i + \hat{\xi}_i = \hat{e}(w \cdot x_i + b - t_i) \ .$$

In second place, note that $\xi_i \hat{\xi}_i = 0$; that is, one of them always has to be null. To have a better understanding of the role of the slack variables and the ϵ-tube we can see an example in Figure 2.3.4 We can interpret this problem then as a regularized regression in which the first term of the objective function is the errors made and the second term is the regularization. Just as before, the parameter C tunes the relevance of both terms. This is very similar to Ridge Regression, in which we try to minimize a function with an error and a regularization term. The main difference is the advantages of the ϵ-insensitive error function chosen over the quadratic error.

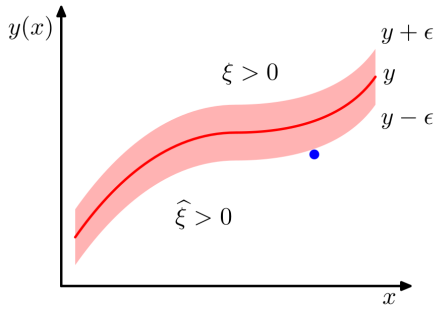

Figure 2.3.4: Image from Bishop's book [10, Chapter 7.1]: The regressor plane is shown in red as well as the ϵ-tube. Slack variables $\xi, \hat{\xi}$ are also represented. Points x_i above the ϵ-tube have $\xi > 0$, points inside the tube have $\xi = \hat{\xi} = 0$, points below the ϵ-tube $\hat{\xi} > 0$.

2.3.2 Analysis

In Subsection 2.3.1 we have shown the problems that motivate the SVMs. A similar optimization problem is obtained both for regression and classification. We can define the following concepts now.

Definition 2.3.1 (SVC Problem)**.** *Given a classification problem*

$$\{(x_1, l_1), \ldots, (x_M, l_M)\}\ , x_i \in \mathcal{X} \subset \mathbb{R}^d\ , l \in \{-1, 1\}\ ,$$

the corresponding optimization problem is

$$\begin{aligned} \underset{w,b,\xi}{\arg\min} \quad & J(w, b, \xi) = C \sum_{i=1}^{M} \xi_i + \frac{1}{2}||w||^2 \\ s.t. \quad & l_i(w \cdot x_i + b) \geq 1 - \xi_i,\ i = 1, \ldots, M, \\ & \xi_i \geq 0,\ i = 1, \ldots, M\ . \end{aligned} \tag{2.3.4}$$

Definition 2.3.2 (SVR Problem)**.** *Given a regression problem,*

$$\{(x_1, t_1), \ldots, (x_M, t_M)\}\ , x_i \in \mathcal{X} \subset \mathbb{R}^d\ , t_i \in \mathbb{R}\ ,$$

the corresponding optimization problem is

$$\begin{aligned} \underset{w,b,\xi,\hat{\xi}}{\arg\min} \quad & J(w, \xi) = C \sum_{i=1}^{M} (\xi_i + \hat{\xi}_i) + \frac{1}{2}||w||^2 \\ s.t. \quad & w \cdot x_i + b \geq t_i - \epsilon - \xi_i,\ i = 1, \ldots, M\ , \\ & w \cdot x_i + b \leq t_i + \epsilon + \hat{\xi}_i,\ i = 1, \ldots, M\ , \\ & \xi_i, \hat{\xi}_i \geq 0,\ i = 1, \ldots, M\ . \end{aligned} \tag{2.3.5}$$

Both problems are similar but have some differences too. However, it is possible to define a formulation that unify both problems. We first show the unifying formulation.

Definition 2.3.3 (Primal SVM Problem)**.** *Given a sample*

$$\{(x_1, y_1, p_1), \ldots, (x_N, y_N, p_N)\}\ , x_i \in \mathcal{X} \subset \mathbb{R}^d\ , y_i \in \mathbb{R}\ , p_i \in \mathbb{R}\ ,$$

a SVM problem is the following optimization problem,

$$\begin{aligned} \underset{w,b,\xi}{\arg\min} \quad & J(w,b,\xi) = C\sum_{n=1}^{N}\xi_i + \frac{1}{2}||w||^2 \\ s.t. \quad & y_i(w\cdot x_i + b) \geq p_i - \xi_i,\ i = 1,\dots,N, \\ & \xi_i \geq 0,\ i = 1,\dots,N\ . \end{aligned} \tag{2.3.6}$$

With this definition it is easy to check the following proposition.

Proposition 2.3.1. *The problem* (2.3.6) *is equivalent to* (2.3.4) *when choosing* $M = N$ *and*

$$y_i = t_i,\ p_i = 1\ i = 1,\dots,N\ ;$$

and it is also equivalent to (2.3.5) *when we choose* $N = 2M$ *and we select*

$$\begin{aligned} y_i = 1,\ p_i = t_i - \epsilon,\ i = 1,\dots,M \\ y_{M+i} = -1,\ p_{M+i} = -t_i - \epsilon,\ i = 1,\dots,M\ . \end{aligned}$$

The unified problem defined in (2.3.6) is advantageous because we can use it to make an analysis that is valid for both SVC and SVR at the same time. We have reached a point then, where we have a single optimization problem, to which we can apply the optimization theory previously explained. In first place, notice that the problem (2.3.6) has a convex objective function and affine restrictions; thus, we can apply the Strong Duality Theorem 2.2.4. It is interesting then to obtain its dual problem, and in consequence, its Lagrangian. The **Lagrangian** of the primal SVM problem is

$$\begin{aligned} \mathcal{L}(w,b,\xi,\alpha,\beta) = & \frac{1}{2}||w||^2 + C\sum_{i=1}^{N}\xi_i \\ & - \sum_{i=1}^{N}\alpha_i\left(y_i(w\cdot x_i + b) - p_i + \xi_i\right) - \sum_{i=1}^{N}\beta_i\xi_i, \\ & \alpha,\beta \geq 0\ , \end{aligned} \tag{2.3.7}$$

where

$$\alpha = (\alpha_1,\dots,\alpha_N)\ ,\ \beta = (\beta_1,\dots,\beta_N)\ .$$

Since we want to obtain the dual function, that is

$$\Theta(\alpha,\beta) = \min_{w,b,\xi}\mathcal{L}(w,b,\xi,\alpha,\beta)\ ,$$

we have to compute the following derivatives:

$$\frac{\partial\mathcal{L}}{\partial w} = w - \sum_{i=1}^{N}\alpha_i y_i x_i \tag{2.3.8}$$

$$\frac{\partial\mathcal{L}}{\partial b} = \sum_{i=1}^{N}\alpha_i y_i \tag{2.3.9}$$

$$\frac{\partial\mathcal{L}}{\partial\xi_i} = C - \alpha_i - \beta_i \tag{2.3.10}$$

Then, making $\dfrac{\partial \mathcal{L}}{\partial w} = 0$ we get

$$w^* = \sum_{i=1}^{N} \alpha_i y_i x_i \ , \tag{2.3.11}$$

which gives us a relation between the primal variable w and the dual variable α. Making $\dfrac{\partial \mathcal{L}}{\partial b} = 0$ we get

$$\sum_{i=1}^{N} \alpha_i y_i = 0 \ , \tag{2.3.12}$$

which gives us a condition over the dual variable α, this is called the *equality constraint*. Finally, making $\frac{\partial \mathcal{L}}{\partial \xi_i}$ we have the condition $\alpha_i + \beta_i = C$, which, alongside the feasibility of α and β gives $0 \leq \alpha_i \leq C$, that is called the *equality constraint*. Using this conditions we can write the dual function as follows,

$$\begin{aligned}
\Theta(\alpha) &= \mathcal{L}(w^*, b^*, \xi^*, \alpha, \beta) \\
&= \frac{1}{2}\langle \sum_{i=1}^{N} \alpha_i y_i x_i, \sum_{j=1}^{N} \alpha_i y_j x_j \rangle - \sum_{i=1}^{N} \alpha_i y_i \left(\sum_{j=1}^{N} \alpha_j y_j x_j \right) \cdot x_i + \sum_{i=1}^{N} p_i \\
&= -\frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j x_i \cdot x_j + \sum_{i=1}^{N} \alpha_i p_i \ .
\end{aligned}$$

Then, since the dual function is maximized, the **Dual Problem** is the next one,

$$\begin{aligned}
\min_{\alpha} \quad & \Theta(\alpha) = \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j x_i \cdot x_j - \sum_{i=1}^{N} \alpha_i p_i \\
\text{s.t.} \quad & 0 \leq \alpha_i \leq C, \ i = 1, \ldots, N \ \text{(box constraint)} \ , \\
& \sum_{i=1}^{N} y_i \alpha_i = 0, \ \ \text{(equality constraint)} \ .
\end{aligned} \tag{2.3.13}$$

The objective function $\Theta(\alpha)$ of the Dual Problem is quadratic, hence it is convex on α. Moreover, the constraints over α are affine and simpler than in the Primal Problem. With these properties, it is sensible to say that solving the Dual Problem is an easier task than solving the primal one. Using the Strong Duality Theorem, it is sufficient to find a solution of the dual α^* and then $\Theta(\alpha^*) = f(w^*)$. Moreover, using the equation (2.3.11) we can recover the primal solution from the dual one. To find the solution of the dual problem, we use the KKT Theorem [9], for which we need first to write the KKT conditions, which are

$$\alpha_i \geq 0 \ , \tag{2.3.14}$$

$$y_i(w \cdot x_i + b) - p_i + \xi_i \geq 0 \ , \tag{2.3.15}$$

$$\alpha_i(y_i(w \cdot x_i + b) - p_i + \xi_i) = 0 \ , \tag{2.3.16}$$

$$\beta_i \geq 0 \ , \tag{2.3.17}$$

$$\xi_i \geq 0 \ , \tag{2.3.18}$$

$$\beta_i \xi_i = 0 \ , \tag{2.3.19}$$

$$\alpha_i + \beta_i = C \ . \tag{2.3.20}$$

Notice that α is only present in the three first conditions and the last, while β is present in the last four. And moreover, the dual problem only depends on α. However α and β are connected by Equation (2.3.20); so given α_i^*, the value for β_i^* is $C - \alpha_i^*$. Our goal then, is to find α^* that fulfills the first three KKT conditions. To do so, some algorithms have been developed, the most popular one, named the SMO algorithm, will be presented in the following section. However, before showing the method to find the solution, it is interesting to analyze some properties of this solution. In first place, notice that the primal solution w^* depends on the dual solution in the following way

$$w^* = \sum_{i=1}^{N} y_i \alpha_i^* x_i \ ;$$

however, the conditions (2.3.16) and (2.3.15) implies that any point x_i of the sample such that $y_i(w \cdot x_i + b) - p_i + \xi_i > 0$ has an associated $\alpha_i = 0$ .Then, w^* depends only on a subset of the points $\mathcal{I} = \{i : \ y_i(w \cdot x_i + b) - p_i + \xi_i = 0\}$, that is

$$w^* = \sum_{i \in \mathcal{I}} y_i \alpha_i^* x_i \ .$$

It is clear that the solution of the optimization problem depends only on the vectors x_i such that $i \in I$ and thus $\alpha_i > 0$, these points receive the name of *Support Vectors* and that is the reason these models are named *Support Vector Machines*. It is relevant to study the properties of these support vectors. Using the KKT conditions, we can classify the *Support Vectors* in terms of the value of α_i. This classification is shown in Table 2.3.1.

α_i	β_i	ξ_i	$y_i(w \cdot x_i + b)$
$0 < \alpha_i < C$	$\beta > 0$	$\xi_i = 0$	$y_i(w \cdot x_i + b) = p_i$
$\alpha_i = C$	$\beta_i = 0$	$0 \leq \xi_i < p_i$	$0 \leq y_i(w \cdot x_i + b) \leq p_i$
$\alpha_i = C$	$\beta_i = 0$	$p_i < \xi_i$	$y_i(w \cdot x_i + b) < 0$

Table 2.3.1: Classification of *Support Vectors* in terms of the value of α_i.

In order to interpret Table 2.3.1 we will focus first on classification where $p_i = 1$. In that case, the first row indicates $y_i(w \cdot x_i + b) = 1$; that means that the vector is correctly classified and just on the margin of the model. The second row indicates $0 \leq y_i(w \cdot x_i + b) \leq 1$; that is a point that is correctly classified but inside the margin. Finally, the last row characterizes the vectors that are misclassified. An example is shown in Figure 2.3.5.

The same analysis can be done with the regression problem, for the sake of concision we will see the case where $y_i = 1$ and $p_i = t_i - \epsilon$. In that case the first row characterizes the vectors x_i with $w \cdot x_i + b = ti - \epsilon$; that is, those in the lower margin of the ϵ-tube. The second row is $0 \leq w \cdot x_i + b < t_i - \epsilon$, which are the points below the ϵ-tube. Finally the last row is $w \cdot x_i + b < 0 < t_i - \epsilon$ which are also below the tube. In conclusion, the solution of the SVM optimization problem depends on those points that are "extreme", those that are not easy to classify or to find a regressor plane for.

2.3.3 Kernel Trick

Until now we have seen the SVMs as linear models. Moreover, their initial motivation was to find maximum separability in linearly separable problems. Linear separability has the inconvenient of being unlikely in real world data. However, the following statement based on the Theorem proved by Cover [11] helps in this issue.

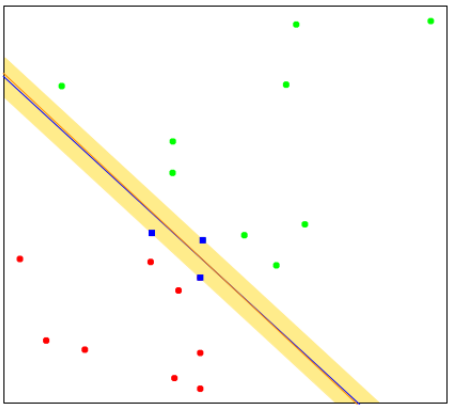

Figure 2.3.5: The blue line shows the optimal separating plane, the shaded region is the maximum separation between the two classes. *Support Vectors* are shown in blue.

A complex pattern-classification problem, cast in a high-dimensional space non-linearly, is more likely to be linearly separable than in a low-dimensional space, provided that the space is not densely populated.

This statement is really useful: we just need to map the original features x_i into a high-dimesional space (even infinite-dimensional) $\mathcal{F}$ through some mapping function $\phi(x)$. Then, the SVM primal problem can be written as

$$\begin{aligned} \underset{w,b,\xi}{\arg\min} \quad & J(w,b,\xi) = C\sum_{n=1}^{N}\xi_i + \frac{1}{2}||w||^2 \\ \text{s.t.} \quad & y_i(w\cdot\phi(x_i)+b) \geq p_i - \xi_i,\ i=1,\dots,N, \\ & \xi_i \geq 0,\ i=1,\dots,N\ , \end{aligned} \tag{2.3.21}$$

where $w \in \mathcal{F}$. However, the complexity and computational cost of solving an optimization problem increases with the dimension of the feature space. Moreover, it would be intractable if $\mathcal{F}$ is infinite-dimensional. Nevertheless, SVMs have the property that their solution can be expressed in terms of the inner products of its patterns; therefore, we can use the kernel-induced feature spaces. This method of implicitly mapping the initial features in a high-dimensional space is called the *Kernel Trick*. In first place we choose a kernel function $k(x,y)$; by Mercer's Theorem 2.1.1 there exists an associated mapping ϕ. Then, the dual problem can be written as follows,

$$\begin{aligned} \min_{\alpha} \quad & \Theta(\alpha) = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j k(x_i,x_j) - \sum_{i=1}^{N}\alpha_i p_i \\ \text{s.t.} \quad & 0 \leq \alpha_i \leq C,\ i=1,\dots,N \text{ (box constraint)}\ , \\ & \sum_{i=1}^{N} y_i\alpha_i = 0, \quad \text{(equality constraint)}\ , \end{aligned} \tag{2.3.22}$$

where $k(x_i,x_j)) = \phi(x_i)\cdot\phi(x_j)$. Moreover, the classification of a point x_j requires computing $w\cdot\phi(x_j)$; using (2.3.11) the solution is the following:

$$w\cdot\phi(x_j) = \sum_{i\in\mathcal{I}} y_i\alpha_i\phi(x_i)\cdot\phi(x_j) = \sum_{i\in\mathcal{I}} y_i\alpha_i k(x_i,x_j)\ .$$

For finding b, we know that for $x_j,\ j \in \mathcal{I}$ such that $\xi_j = 0$ we have $y_j(w \cdot \phi(x_j) + b) = p_j$, then we can get b as

$$b = p_j/y_j - w \cdot \phi(x_i) = p_j/y_j - \sum_{i \in \mathcal{I}} y_i \alpha_i k(x_i, x_j) \ .$$

2.4 SMO Algorithm

In the previous section it has been shown that SVMs have multiple desirable properties such as the maximal separability and the possibility of applying the *kernel trick*. However, it is still necessary to develop a method required for training the SVMs. In this work we will focus on the SMO algorithm [12].

2.4.1 Update Step

In first place, it is important to state that SMO solves the Dual Problem (2.3.13); thus, using (2.3.22) it solves the Primal Problem (2.3.21) as well. That is, using the optimization theory presented in Section 2.2, SMO aims to find

$$\alpha^* = (\alpha_1^*, \alpha_2^*, \ldots, \alpha_N^*)$$

that fulfills the KKT conditions, and therefore is an optimal solution. Then, using

$$w^* = \sum_{i \in \mathcal{I}} y_i \alpha_i^* \phi(x_i) \ ,$$

where $\mathcal{I}$ are the indices of the Support Vectors and

$$b^* = p_j/y_j - \sum_{i \in \mathcal{I}} y_i \alpha_i^* k(x_i, x_j)$$

for any $j \in \mathcal{I}$, it finds the optimal solution to the Primal Problem. In order to find α^* the idea is to begin with an initial $\alpha(0) = \vec{0}$; then we update $\alpha(\tau)$, that is, the vector α at the iteration τ, as follows

$$\alpha(\tau + 1) = \alpha(\tau) + \delta(\tau) \ ,$$

in a way that the update maximizes

$$\Theta^\tau - \Theta^{\tau+1} = \Theta(\alpha(\tau)) - \Theta(\alpha(\tau + 1)) = \Theta(\alpha(\tau)) - \Theta(\alpha(\tau) + \delta(\tau)) \ .$$

In order to find such update $\delta(\tau)$, the easiest solution would be to update a single value α_i at a time; however, the Dual Problem includes the equality constraint

$$\sum_{i=1}^{N} y_i \alpha_i = 0 \ ,$$

and updating a single α_i will break the constraint. For that reason, the simplest update that fulfills the constraint involves two coefficients. That is, first we select U, L with some heuristic rule that we study in Subsection 2.4.2; then, given the indices U, L we make the following update,

$$\begin{aligned} \alpha_U(\tau + 1) &= \alpha_U(\tau) + \delta_U(\tau) \\ \alpha_L(\tau + 1) &= \alpha_L(\tau) + \delta_L(\tau) \ , \end{aligned}$$

and we have to ensure

$$\sum_{i=0}^{N} \alpha_i(\tau+1)y_i = 0 \ .$$

It is possible then to connect $\delta_L(\tau)$ and $\delta_U(\tau)$. To do this we use the equality constraint in times τ and $\tau+1$, which can be written as follows

$$\begin{aligned}(\tau) &: \alpha_U(\tau)y_U + \alpha_L(\tau)y_L + \sum_{i\neq U,L} \alpha_i(\tau)y_i = 0; \\ (\tau+1) &: \alpha_U(\tau+1)y_U + \alpha_L(\tau+1)y_L + \sum_{i\neq U,L} \alpha_i(\tau+1)y_i \\ &= (\alpha_U(\tau)+\delta_U(\tau))y_U + (\alpha_L(\tau)+\delta_L(\tau))y_L + \sum_{i\neq U,L} \alpha_i(\tau+1)y_i = 0 \ .\end{aligned}$$

Therefore, using the first equation and the last one, it is easy to see that

$$\delta_U(\tau)y_U + \delta_L(\tau)y_L = 0 \ . \tag{2.4.1}$$

Using this equation it is possible to express the difference $\Theta^\tau - \Theta^{\tau+1}$ in terms of just δ_L, and we have

$$\begin{aligned}\Theta^\tau - \Theta^{\tau+1} &= \frac{1}{2}\alpha(\tau)^t Q\alpha(\tau) - p\alpha(\tau) - \left(\frac{1}{2}\alpha(\tau+1)^t Q\alpha(\tau+1) - p\alpha(\tau+1)\right) \\ &= \frac{1}{2}\alpha(\tau)^t Q\alpha(\tau) - p\alpha(\tau) \\ &\quad - \left(\frac{1}{2}(\alpha(\tau)+\delta(\tau))^t Q(\alpha(\tau)+\delta(\tau)) - p(\alpha(\tau)+\delta(\tau))\right) \\ &= -\frac{1}{2}\left(\delta(\tau)^t Q\delta(\tau) + \alpha(\tau)^t Q\delta(\tau) + \delta(\tau)^t Q\alpha(\tau)\right) + p\delta(\tau) \\ &= -\frac{1}{2}\delta(\tau)^t Q\delta(\tau) - \left(\delta(\tau)^t Q\alpha(\tau) - p\delta(\tau)\right) \ ,\end{aligned} \tag{2.4.2}$$

where p corresponds to the unifying formulation introduced in (2.3.21). Since it can be expressed with variables corresponding only to time τ, the iteration time can be omitted resulting in the following equation

$$\Theta^\tau - \Theta^{\tau+1} = -\frac{1}{2}\delta^t Q\delta - \left(\delta^t \hat{Q}\alpha - p\delta\right) = -\frac{1}{2}A - B \ . \tag{2.4.3}$$

where

$$\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_N)^t$$

and

$$\delta = (0, \ldots, 0, \delta_L, 0, \ldots, 0, \delta_U, 0 \ldots, 0)^t \ ,$$

where only the positions L and U are not null. Given these clarifications, (2.4.3) can be further expanded. The term denoted as A can be expressed as follows

$$\begin{aligned}A &= \delta^t Q\delta \\ &= \delta_U^2 Q_{UU} + 2\delta_U\delta_L Q_{UL} + \delta_L^2 Q_{LL} \\ &= \delta_L^2\phi(x_U)\cdot\phi(x_U) + 2(-y_U y_L\delta_L)\delta_L y_U y_L\phi(x_U)\cdot\phi(x_L) + \delta_L^2\phi(x_L)\cdot\phi(x_L) \\ &= \delta_L^2\left\|\phi(x_U)\right\|^2 - 2\delta_L^2\phi(x_U)\cdot\phi(x_L) + \delta_L^2\left\|\phi(x_L)\right\|^2 \\ &= \delta_L^2\left\|\phi(x_U) - \phi(x_L)\right\|^2 \ .\end{aligned}$$

Besides, we can develop the term B in the following way,

$$\begin{aligned} B &= \delta Q\alpha - p\delta = \delta_L \left((Q\alpha)_L - p_L\right) + \delta_U \left((Q\alpha)_U - p_U\right) \\ &= \delta_L \left(\sum_{i=1}^{N} Q_{Li}\alpha_i - p_L\right) + \delta_U \left(\sum_{i=1}^{N} Q_{Ui}\alpha_i - p_U\right) \\ &= \delta_L \left(\sum_{i=1}^{N} Q_{Li}\alpha_i - p_L\right) - y_U y_L \delta_L \left(\sum_{i=1}^{N} Q_{Ui}\alpha_i - p_U\right) \\ &= y_L \delta_L \left(y_L \left(\sum_{i=1}^{N} Q_{Li}\alpha_i - p_L\right) - y_U \left(\sum_{i=1}^{N} Q_{Ui}\alpha_i - p_U\right)\right) \ . \end{aligned}$$

Moreover, the gradient of the dual objective function is the following

$$\nabla\Theta(\alpha) = Q\alpha - p \ ; \tag{2.4.4}$$

therefore the position j of the gradient is

$$\nabla\Theta(\alpha)_j = Q_j\alpha - p_j = \left(\sum_{i=1}^{N} Q_{ji}\alpha_i - p_j\right) \ .$$

Using this, the following is true:

$$B = y_L \delta_L \left(y_L (\nabla\Theta)_L - y_U (\nabla\Theta)_U\right) \ .$$

Finally, using the results for A and B, the difference of the dual objective function can be expressed as follows:

$$\begin{aligned} \Theta^{\tau} - \Theta^{\tau+1} &= -y_L \delta_L \left(y_L (\nabla\Theta)_L - y_U (\nabla\Theta)_U\right) - \frac{1}{2}\delta_L^2 \left\|\phi(x_U) - \phi(x_L)\right\|^2 \\ &= -\psi(\delta_L) \ . \end{aligned} \tag{2.4.5}$$

With this definition, the goal is to minimize $\psi(\delta_L)$, and since it is a convex function it is sufficient to find δ_L^* such that $\psi'(\delta_L^*) = 0$ where

$$\psi'(\delta_L) = y_L \left(y_L (\nabla\Theta)_L - y_U (\nabla\Theta)_U\right) + \delta_L \left\|\phi(x_U) - \phi(x_L)\right\|^2 \ .$$

The result is the following

$$\delta_L^* = \frac{y_L \left(y_U (\nabla\Theta)_U - y_L (\nabla\Theta)_L\right)}{\left\|\phi(x_U) - \phi(x_L)\right\|^2} \ . \tag{2.4.6}$$

Defining $\bar{\lambda}$ as

$$\bar{\lambda} = \frac{\left(y_U (\nabla\Theta)_U - y_L (\nabla\Theta)_L\right)}{\left\|\phi(x_U) - \phi(x_L)\right\|^2} \ ,$$

then the update of the dual variables α_U and α_L would the following

$$\begin{aligned} \alpha_L^{\tau+1} &= \alpha_L^{\tau} + y_L \bar{\lambda} \\ \alpha_U^{\tau+1} &= \alpha_U^{\tau} - y_U \bar{\lambda} \ ; \end{aligned}$$

however, it is necessary to clip $\bar{\lambda}$ in order to keep the dual variables in the feasible region, that is $0 \leq \alpha_U, \alpha_L \leq C$. To achieve this, the update is defined in the following way

$$\begin{aligned} &\lambda = \min\left(C - \alpha_L^{\tau}, \bar{\lambda}\right) \text{ if } y_L = 1 \ , \\ &\lambda = \min\left(\alpha_L^{\tau}, \bar{\lambda}\right) \text{ if } y_L = \text{-}1 \ , \\ &\lambda = \min\left(\alpha_U^{\tau}, \bar{\lambda}\right) \text{ if } y_U = 1 \ , \\ &\lambda = \min\left(C - \alpha_U^{\tau}, \bar{\lambda}\right) \text{ if } y_U = \text{-}1 \ . \end{aligned}$$

Then, we use λ to update the dual variables

$$\alpha_L^{\tau+1} = \alpha_L^\tau + y_L\lambda \,,$$
$$\alpha_U^{\tau+1} = \alpha_U^\tau - y_U\lambda \,.$$

It can be seen that even with the clipping, the update makes Θ smaller. To check just one case, set $y_L = -1$ and $\alpha_L^\tau < \bar{\lambda}$, then $\delta_L^\tau = y_L\alpha_L^\tau$, therefore

$$\begin{aligned}\Theta^\tau - \Theta^{\tau+1} = \psi(y_L\alpha_L^\tau) &= \alpha_L^\tau \left(y_U(\nabla\Theta)_U - y_L(\nabla\Theta)_L\right) - \frac{1}{2}\left(\alpha_L^\tau\right)^2 \|\phi(x_U) - \phi(x_L)\|^2 \\ &\geq \alpha_L^\tau \left(y_U(\nabla\Theta)_U - y_L(\nabla\Theta)_L\right) > 0 \,.\end{aligned}$$

The last inequality has not been seen yet since it is dependent on the election of U and L. In the next subsection a heuristic rule is developed to select U and L, and moreover we prove the last inequality.

2.4.2 Selection Step: Selecting the Optimal (U, L)

Before presenting the heuristic methods that are used in the selection step it is important to recall the KKT conditions of this optimization problem, which are

$$\begin{aligned}\alpha_i &\geq 0 \,,\\ y_i(w\cdot\phi(x_i) + b) - p_i + \xi_i &\geq 0 \,,\\ \alpha_i(y_i(w\cdot\phi(x_i) + b) - p_i + \xi_i) &= 0 \,,\\ \beta_i &\geq 0 \,,\\ \xi_i &\geq 0 \,,\\ \beta_i\xi_i &= 0 \,,\\ \alpha_i + \beta_i &= C \,.\end{aligned}$$

The KKT conditions are important because given a tuple $(w^*, b^*, \xi^*, \alpha^*, \beta^*)$ that fulfills the KKT conditions, then the tuple (w^*, b^*, ξ^*) is the optimal solution of the primal and (α^*, β^*) is the optimal solution of the dual. In order to have an easier condition to check optimality the following Lemma can be proved.

Lemma 2.4.1. *Define the following sets*

$$\begin{aligned}I_U^+ &= \{i : y_i = 1 \wedge \alpha_i > 0\} \,,\ I_U^- &= \{i : y_i = -1 \wedge \alpha_i < C\} \,,\\ I_L^+ &= \{i : y_i = 1 \wedge \alpha_i < C\} \,,\ I_L^- &= \{i : y_i = -1 \wedge \alpha_i > 0\} \,,\end{aligned}$$

and also define

$$I_U = I_U^+ \cup I_U^- \,,$$
$$I_L = I_L^+ \cup I_L^- \,;$$

then, if the KKT conditions hold,

$$w\cdot\phi(x_u) - y_u p_u \leq -b \leq w\cdot\phi(x_l) - y_l p_l,\ \forall u \in I_U,\ \forall l \in I_L \,. \tag{2.4.7}$$

Proof. The two cases $0 \leq \alpha_i < C$ and $0 < \alpha_i \leq C$ are proved separately. Note that although the two sets are not disjoint, every α_i is covered in one of them. The first case is the following: $\alpha_i < C \implies \beta_i > 0 \implies \xi_i = 0$, which can be followed from the KKT conditions; then

$$y_i\left(w\cdot\phi(x_i) - y_i p_i + b\right) - p_i \geq 0 \,;$$

that is

$$w \cdot \phi(x_i) - y_i p_i \geq -b \text{ if } y_i = 1 \ (I_L^+) \ ,$$
$$w \cdot \phi(x_i) - y_i p_i \leq -b \text{ if } y_i = -1 \ (I_U^-) \ .$$

The second case is the following: $\alpha_i > 0 \implies y_i \left(w \cdot \phi(x_i) - y_i p_i + b\right) - p_i + \xi_i = 0$, which can be followed from the KKT conditions; then

$$y_i \left(w \cdot \phi(x_i) - y_i p_i + b\right) - p_i \leq 0 \ ;$$

that is

$$w \cdot \phi(x_i) - y_i p_i \leq -b \text{ if } y_i = 1 \ (I_U^+) \ ,$$
$$w \cdot \phi(x_i) - y_i p_i \geq -b \text{ if } y_i = -1 \ (I_L^-) \ .$$

□

This lemma provides an easy two-step rule for selecting (U^*, L^*); that is, the indices to update. The rule consists on selecting the pair of indices (U, L) that violate the most the KKT conditions. More formally, notice first the following:

$$\Theta(\alpha) = \frac{1}{2}\langle w^*, \sum_{i=1}^{N} \alpha_i y_i \phi(x_i)\rangle - \sum_{i=1}^{N} \alpha_i p_i \ ;$$

then taking derivatives with respect to α_i the result is

$$(\nabla\Theta(\alpha))_j^r = \frac{\partial\Theta(\alpha)}{\partial\alpha_i} = \langle w^*, y_i\phi(x_i)\rangle - p_i \ .$$

Therefore, multiplying the partial derivative times y_i we get

$$y_i(\nabla\Theta(\alpha))_j^r = y_i\frac{\partial\Theta(\alpha)}{\partial\alpha_i} = \langle w^*, \phi(x_i)\rangle - y_i p_i \ ;$$

thus, the condition of the equation (2.4.7) can be written as

$$y_u(\nabla\Theta(\alpha))_u \leq y_l(\nabla\Theta(\alpha))_l, \ \forall u \in I_U, \ \forall l \in I_L \ .$$

Given this result, we define Δ as

$$\Delta(\alpha) = \max_{u \in I_U}(y_u(\nabla\Theta)_u) - \min_{l \in I_L}(y_l(\nabla\Theta)_l) \ ;$$

this definition implies that if $\Delta(\alpha) \leq 0$, then by the Lemma 2.4.1, α is the optimal solution. However, if $\Delta(\alpha) > 0$ then the KKT conditions are not met and therefore α is not optimal. Moreover any pair $u \in I_U$, $l \in I_L$ such that

$$y_u(\nabla\Theta(\alpha))_u - y_l(\nabla\Theta(\alpha))_l > 0$$

is called a 'violating pair'. With this information, the heuristic rule used is to select U^*, L^* in the following way:

$$\begin{aligned} U^* &= \arg\max_{u \in I_U}(y_u(\nabla\Theta)_u) \ , \\ L^* &= \arg\min_{l \in I_L}(y_l(\nabla\Theta)_l) \ . \end{aligned} \tag{2.4.8}$$

Given the definition of the selection rule, it shows that the indices selected are those that most violate the KKT conditions and thus the rule is called 'maximal violating pair' rule.

The first part of this subsection is focused on the definition of 'violating pair' and moreover, it provides the motivation for the 'maximal violating pair' rule. However, a better selection can be made. In first place, recall that the main goal to achieve a faster convergence is to minimize

$$\Psi(\delta) = \Delta\Theta(\alpha) = \Theta(\alpha+\delta) - \Theta(\alpha) \ ,$$

which, using (2.4.2), can be written as

$$\begin{aligned}\Psi(\delta) = \Delta\Theta(\alpha) &= \Theta(\alpha+\delta) - \Theta(\alpha) \\ &= \frac{1}{2}\delta(\tau)^t Q\delta(\tau) - \left(\delta(\tau)^t Q\alpha(\tau) + p\delta(\tau)\right) \ .\end{aligned}$$

Moreover, using the derivatives of $\Theta(\alpha)$ which are

$$\begin{aligned}\nabla\Theta(\alpha) &= \alpha^t Q + p \ , \\ \nabla^2\Theta(\alpha) &= Q \ ,\end{aligned}$$

we can write $\Psi(\delta)$, which is quadratic, in the following way

$$\Psi(\delta) = \Delta\Theta(\alpha) = \Theta(\alpha+\delta) - \Theta(\alpha) = \nabla\Theta(\alpha)\delta + \frac{1}{2}\delta^t\nabla^2\Theta(\alpha)\delta \ .$$

We have already seen in (2.4.5) that it can be expressed just in terms of δ_L:

$$\Psi(\delta) = \psi(\delta_L) = y_L\delta_L\left(y_L(\nabla\Theta)_L - y_U(\nabla\Theta)_U\right) + \frac{1}{2}\delta_L^2\left(\|\phi(x_U) - \phi(x_L)\|^2\right) \ .$$

It is clear that although the function to optimize is dependent on the first and second order derivatives of $\Theta(\alpha)$, with the *maximal violating pair rule* we are only using information about the first derivative $\nabla\Theta(\alpha)$. Nevertheless, Equation (2.4.6) gives us a closed solution for the optimal update δ^* in terms of (U, L); then, our goal is to choose (U, L) that minimize $\Psi(\delta^*)$. To do that the following proposition can be used.

Proposition 2.4.1. *Let (U, L) be a violating pair, then $\Psi(\delta)$ has the optimal value*

$$-\frac{1}{2}\frac{\left(y_U(\nabla\Theta)_U - y_L(\nabla\Theta)_L\right)^2}{\|\phi(x_U) - \phi(x_L)\|^2} \ . \tag{2.4.9}$$

Proof. With the optimal δ_L^* obtained in (2.4.6) we just need to compute $\Psi(\delta^*) = \psi(\delta_L^*)$,

$$\begin{aligned}\psi(\delta_L^*) &= y_L\delta_L^*\left(y_L(\nabla\Theta)_L - y_U(\nabla\Theta)_U\right) + \frac{1}{2}(\delta_L^*)^2\left(\|\phi(x_U) - \phi(x_L)\|^2\right) \\ &= y_L\left(\frac{y_L\left(y_U(\nabla\Theta)_U - y_L(\nabla\Theta)_L\right)}{\|\phi(x_U) - \phi(x_L)\|^2}\right)\left(y_L(\nabla\Theta)_L - y_U(\nabla\Theta)_U\right) + \\ &\quad \frac{1}{2}\left(\frac{y_L\left(y_U(\nabla\Theta)_U - y_L(\nabla\Theta)_L\right)}{\|\phi(x_U) - \phi(x_L)\|^2}\right)^2\left(\|\phi(x_U) - \phi(x_L)\|^2\right) \\ &= -\frac{1}{2}\left(\frac{y_L\left(y_U(\nabla\Theta)_U - y_L(\nabla\Theta)_L\right)^2}{\|\phi(x_U) - \phi(x_L)\|^2}\right) \ .\end{aligned}$$

□

Using Proposition 2.4.1 it is sufficient to choose (U, L) as a violating pair that minimizes (2.4.9). However, this method requires a quadratic search. The solution, called *Second Order SMO* [13], is to select a violating pair (U, L) in the following way:

$$\begin{aligned} U^* &= \arg\max_{u \in I_U} (y_u (\nabla\Theta)_u) \ , \\ L^* &= \arg\min_{l \in I_L} \left(-\frac{(y_U (\nabla\Theta)_U - y_L (\nabla\Theta)_L)^2}{\|\phi(x_U) - \phi(x_L)\|^2} \right) \ . \end{aligned} \tag{2.4.10}$$

By selecting (U, L) in this way, the cost is linear since both steps consist of a linear cost search.

Both selections, (2.4.8) and (2.4.10), make use of the information of the gradient. It is clear then, that it is necessary to maintain the entire gradient $\nabla\Theta$ in order to perform the indices selection. To do this, we use the following equation,

$$\nabla\Theta(\alpha)_j = Q_j \alpha - p_j = \left(\sum_{i=1}^{N} Q_{ji} \alpha_i - p_j \right) \ .$$

Then, the initial gradient is $(\nabla\Theta)^0 = \vec{0}$, and at iteration τ we have the gradient

$$\nabla\Theta(\alpha)_j^\tau = Q_j \alpha^\tau - p_j = \left(\sum_{i=1}^{N} Q_{ji} \alpha_i^\tau - p_j \right) \ ,$$

where Q_j is the j-th row of Q. Using this gradient we perform the selection step at iteration $\tau + 1$ obtaining U, L and then we perform the update step, obtaining $\alpha_U(\tau + 1), \alpha_L(\tau + 1)$. Then, we can express the update of the gradient in a simple way. We first compute the difference between $\nabla\Theta(\alpha)^{\tau+1}$ and $\nabla\Theta(\alpha)^\tau$, that is,

$$\nabla\Theta(\alpha)^{\tau+1} - \nabla\Theta(\alpha)^\tau = \sum_{i=1}^{N} Q_{ji} \delta_i^\tau = Q_{jU} \delta_U^\tau + Q_{jL} \delta_L^\tau \ .$$

Then, we have the following difference,

$$\nabla\Theta(\alpha)^{\tau+1} - \nabla\Theta(\alpha)^\tau = Q\delta^\tau \ ;$$

therefore, to update the gradient we just have to make

$$\nabla\Theta(\alpha)^{\tau+1} = \nabla\Theta(\alpha)^\tau + Q\delta^\tau \ .$$

2.4.3 Computational Cost

With both the update and selection step explained, it is necessary to write the whole algorithm and analyze its computational cost. To do so, *Second Order SMO* is given in Algorithm 1. In order to get the computational cost of the *Second Order SMO* algorithm we will study the cost of each iteration. In first place, the selection step consists on two linear cost searches, that have a $O(2N)$ cost. Then, the Update Step requires just some comparisons and has cost $O(1)$. Also, the Gradient Update requires the multiplication of the matrix Q and the vector δ, where δ is null except for the positions U, L; that means that it has a cost $O(2N)$. Finally, the computation of Δ requires to check the conditions of I_U and I_L for the recently updated variables α_U and α_L, that is, a $O(1)$ cost. Therefore, putting it all together, the computational cost of each iteration of SMO

Algorithm 1: SMO

Data: matrix Q, vector p, parameter C
Result: vector α^*
$\alpha \leftarrow \vec{0}$;
$\nabla\Theta \leftarrow -p$;
$I_U \leftarrow \{i : y_i = 1 \wedge \alpha_i > 0\} \cup \{i : y_i = -1 \wedge \alpha_i < C\}$;
$I_L \leftarrow \{i : y_i = 1 \wedge \alpha_i < C\} \cup \{i : y_i = -1 \wedge \alpha_i > 0\}$;
$\Delta = \max_{u \in I_U}(y_u(\nabla\Theta)_u) - \min_{l \in I_L}(y_l(\nabla\Theta)_l)$; `// KKT conditions`
while $\Delta > tol$ **do**
  $U = \arg\max_{u \in I_U}(y_u(\nabla\Theta)_u)$; `// Selection Step`
  $L = \arg\min_{l \in I_L}\left(-\frac{\left(y_U(\nabla\Theta)_U - y_l(\nabla\Theta)_l\right)^2}{Q_{UU}-2Q_{Ul}+Q_{ll}}\right)$; `// Selection Step`
  $\lambda \leftarrow \frac{\left(y_U(\nabla\Theta)_U - y_L(\nabla\Theta)_L\right)}{Q_{UU}-2Q_{UL}+Q_{LL}}$;
  if $y_L == 1$ **then** $\lambda = \min(C - \alpha_L, \lambda)$;
  else $\lambda = \min(\alpha_L, \lambda)$;
  if $y_U == 1$ **then** $\lambda = \min(\alpha_U, \lambda)$;
  else $\lambda = \min(C - \alpha_U, \lambda)$;
  $\alpha_L \leftarrow \alpha_L + y_L\lambda$; $\alpha_U \leftarrow \alpha_U - y_U\lambda$; `// Update Step`
  $\delta = \vec{0}$;
  $\delta[L] \leftarrow y_L\lambda$; $\delta[U] \leftarrow -y_U\lambda$;
  $\nabla\Theta \leftarrow \nabla\Theta + Q\delta$; `// Gradient Update`
  $I_U \leftarrow \{i : y_i = 1 \wedge \alpha_i < C\} \cup \{i : y_i = -1 \wedge \alpha_i < C\}$;
  $I_L \leftarrow \{i : y_i = 1 \wedge \alpha_i > 0\} \cup \{i : y_i = -1 \wedge \alpha_i > 0\}$;
  $\Delta \leftarrow \max_{u \in I_U}(y_u(\nabla\Theta)_u) - \min_{l \in I_L}(y_l(\nabla\Theta)_l)$; `// KKT conditions`

is $O(2N)$. Moreover, in [14] it has been proven that with this Selection Step the, after a some finite number of iterations, the convergence of the algorithm is linear. In order to get a solution we need to find a number of support vectors that is tipically proportional to the size of the sample; therefore, updating two variables per iteration, at least $O(N)$ iterations are necessary. Since the cost of each iteration is also linear, it is said that the total cost of *Second Order SMO* is larger than a cost $O(N^2)$, and sometimes it is said to have a cost $O(N^{2+\epsilon})$.

Chapter 3

Multitask Learning

In many practical situations a number of statistical models has to be obtained from data for different problems. In the literature each one of these problems is called a task. A classical example used in [1, 4] is the prediction of the marks of students belonging to different schools; the students of each school comprise a different task. Another example is the prediction of wind energy production in different wind farms. In this case the prediction of the production in each wind farm can be seen as a different task. In multiple occasions the tasks are related to one another and in these cases it can be advantageous to learn all tasks simultaneously. Modelling all tasks at once instead of seeing them as independent problems can help to transfer useful information among the different tasks. This global point of view receives the name of multi-task learning. This work has its focus on Support Vector Machines, which are traditionally single-task models. It will be shown how to adapt these models in order to use them in multi-task problems. Moreover, the motivation behind the adaptions will be presented and its consequences will be analyzed with detail. Finally, the algorithms and methods to make a practical use of these multi-task SVMs will be explained.

3.1 Linear Support Vector Machines for Multitask Learning

The first approach of multi-task learning in an SVM context that will be revised is the Regularized Multitask Learning [1], which is also referred to as rMTL. In this work the classical SVM primal problem is adapted to match a multi-task learning framework. In order to do so, one model is obtained for each task. These models are divided into a global or common part and a specific part. The common part of the models tries to capture global properties shared across all the tasks while the specific part tries to capture local aspects of each task. Since the objective function of an SVM model includes a regularization term, both common and specific parts are included in this regularization, each one with its own tuning parameter. Before going into details some clarifications have to be made. In first place, note that an SVM model is defined by its weight vector w and its bias b. However in this algorithm the bias is assumed to be zero, so it will be omitted. Aditionally, as a first approach, in rMTL we assume a linear kernel. Both the inclusion of a bias and the use of different kernels will be treated in posterior sections. Given these considerations, and provided that we have T different tasks, it is necessary to estimate T weight vectors $v_1, \ldots, v_T$. Besides, in relation with this multi-task method, the Bayesian approach [15] assumes that each v_r is a gaussian with unknown mean w and a covariance matrix Σ. That is

$$v_r \sim \mathcal{N}(w, \Sigma),\ r = 1, \ldots, T\ .$$

As an approximation of this idea, in Regularized Multitask Learning it is assumed that there is a common part shared across all models w, and each task has its own "deviation from the mean" represented as w_r. The weight vector v_r is then expressed as

$$v_r = w + w_r,\ r = 1, \dots, T\ , \tag{3.1.1}$$

where w_r is small if every task is very similar, and w_r is large if the tasks are different. Besides this, in order to use this approach it is necessary to know the task r to which each example x corresponds. Therefore, the samples that are used have the following form:

$$\left\{((x_1, 1), y_1^1), \dots, ((x_{m_1}, 1), y_{m_1}^1), \dots, ((x_1, T), y_1^t), \dots, ((x_{m_T}, T), y_{m_T}^T)\right\}\ ,$$

where (x_i, r) represents the i-th example of the task r, y_i^r is the target of the i-th example of the task r and m_r is the number of examples for task r. This notation is complex and it is necessary just when it is interesting to represent the data as a pair (x, t), splitting the data itself and the task. However to simplify the notation, the following representation will be used in most cases:

$$\left\{(x_1^1, y_1^1), \dots, (x_{m_1}^1, y_{m_1}^1), (x_1^2, y_1^2), \dots, (x_{m_2}^2, y_{m_2}^2), \dots, (x_1^T, y_1^T), \dots, (x_{m_T}^T, y_{m_T}^T)\right\}\ .$$

Given this preliminary clarifications, the primal problem or Regularized Multi-task Learning can be stated as

$$\begin{aligned} \underset{w, w_r, \xi}{\arg\min} \quad & J(w, w_r, \xi) = \sum_{r=1}^{T} \sum_{i=1}^{m_r} \xi_i^r + \frac{\lambda_1}{T} \sum_{r=1}^{T} \|w_r\|^2 + \lambda_2 \|w\|^2 \\ \text{s.t.} \quad & y_i^r (w \cdot x_i^r + w_r \cdot x_i^r) \geq p_i^r - \xi_i^r,\ i = 1, \dots, m_r;\ r = 1, \dots, T, \\ & \xi_i^r \geq 0,\ i = 1, \dots, m_r;\ r = 1, \dots, T\ , \end{aligned} \tag{3.1.2}$$

where p_i^r are the coefficients p presented in the second chapter to unify SVC and SVR under the same notation. It is also relevant to notice that the bias is omitted and hence it is assumed to be null. This is a necessary assumption for the solution method developed in following subsections. The λ_1 and λ_2 parameters model the importance given to each part, common or specific, in the regularization. The first intuition is that if λ_1 is much larger than λ_2, then the "deviations" will be more penalized, therefore making every model very similar to the others. On the other side, if λ_1 is much smaller than λ_2, the common part will be penalized resulting in models that are almost independent as they will not share a common part. However, in order to give a deeper insight in the implications of these parameters, a theoretical analysis is carried out in the next subsection. Additionally, it is necessary to know how to get a solution of this problem. A practical method that solves this multi-task learning problem is also given at the end of Subsection 3.1.2

3.1.1 Analysis

The formulation of the primal problem for multi-task SVM defined in (3.1.2) presents some differences to the the classical SVM primal problem. First of all, the objective function depends on $T + 1$ weight vectors, as opposed to the single-task approach when it depends on a single weight vector. Secondly, the regularization term is split in two terms with two different parameters, one for the specific parts, regulated by λ_1, and one for the common part, regulated by λ_2. The traditional primal problem has only one parameter C that controls the relative importance between the errors made and the margin of the model. The single-task problem can be rewritten in a way such that $\lambda = \frac{1}{2C}$ is used to control the

importance of the regularization term, that is, the margin. This version with λ looks more similar to the multi-task formulation of the problem. Moreover, it also has another aspect present in the single-task problem, the fact that there is a restriction for every example in the sample, and each restriction includes an error term ξ_i^r that is positive and measures the error made in the example x_i^r. With the goal of giving a new perspective to the primal problem and achieve a better understanding of its implications, some theoretical results have been proven [1]. In order to shed some light on the primal problem defined in (3.1.2), it is rewritten in a way that is easier to analyze.

Lemma 3.1.1. *Let w^* y w_r^* be solutions of the rMTL primal problem, and set $v_r^* := w^* + w_r^*$, then we get the following equality*

$$w^* = \frac{\lambda_1}{\lambda_1 + \lambda_2} \frac{1}{T} \sum_{r=1}^{T} v_r^* \, . \tag{3.1.3}$$

Proof. Writing the Lagrangian corresponding to the problem defined in (3.1.2) we get

$$\begin{aligned} \mathcal{L}(w, w_r, \xi, \alpha, \gamma) = J(w, w_r, \xi) \\ - \sum_{r=1}^{T} \sum_{i=1}^{m_r} \{\alpha_i^r [y_i^r ((w + w_r) \cdot x_i^r) - p_i^r + \xi_i^r]\} \\ - \sum_{r=1}^{T} \sum_{i=1}^{m_r} \gamma_i^r \xi_i^r \, . \end{aligned}$$

Making $\frac{\partial \mathcal{L}}{\partial w} = 0$, the following result is obtained

$$w^* = \frac{1}{2\lambda_2} \sum_{r=1}^{T} \sum_{i=1}^{m_r} \alpha_i^r y_i^r x_i^r \, . \tag{3.1.4}$$

Also making $\frac{\partial \mathcal{L}}{\partial w_r} = 0$, we obtain

$$w_r^* = \frac{T}{2\lambda_1} \sum_{i=1}^{m_r} \alpha_i^r y_i^r x_i^r \, . \tag{3.1.5}$$

Using (3.1.4) and (3.1.5)

$$w^* = \frac{\lambda_1}{T\lambda_2} \sum_{r=1}^{T} w_r^* \, ;$$

and substituting w_r^* using $v_r^* = w^* + w_r^*$, this is

$$\begin{aligned} w^* = \frac{\lambda_1}{T\lambda_2} \sum_{r=1}^{T} (v_r^* - w^*) \implies \left(1 + \frac{\lambda_1}{\lambda_2}\right) w^* = \frac{\lambda_1}{\lambda_2} \frac{1}{T} \sum_{r=1}^{T} v_r^* \implies \\ \implies \frac{\lambda_2 + \lambda_1}{\lambda_2} w^* = \frac{\lambda_1}{\lambda_2} \frac{1}{T} \sum_{r=1}^{T} v_r^* \implies w^* = \frac{\lambda_1}{\lambda_1 + \lambda_2} \frac{1}{T} \sum_{r=1}^{T} v_r^* \, . \end{aligned}$$

□

This lemma connects the common part within all models and each particular model in the sense that the common part w^* is a fraction of the mean of the particular models where this fraction is determined by λ_1 and λ_2. Using this result the deviations w_r can be eliminated from (3.1.2), thus getting a problem that is dependent only on the real models v_r, without taking into account a common and a specific part. The next lemma makes this explicit.

Lemma 3.1.2. *The solution of the problem defined in* (3.1.2) *is equivalent to the solution of the following problem:*

$$\begin{aligned} \underset{v_r,\xi}{\arg\min} \quad & J(v_r,\xi) = \sum_{r=1}^{T}\sum_{i=1}^{m_r}\xi_i^r + \rho_1\sum_{t=1}^{T}\|v_t\|^2 + \rho_2\sum_{r=1}^{T}\left\|v_r - \frac{1}{T}\sum_{s=1}^{T}v_s\right\|^2 \\ s.t. \quad & y_i^r(v_r\cdot x_i^r) \geq p_i^r - \xi_i^r,\ i=1,\dots,m_r;\ r=1,\dots,T\ , \\ & \xi_i^r \geq 0,\ \ i=1,\dots,m_r;\ r=1,\dots,T\ , \end{aligned} \tag{3.1.6}$$

when choosing $\rho_1 = \frac{1}{T}\frac{\lambda_1\lambda_2}{\lambda_1+\lambda_2}$ *and* $\rho_2 = \frac{1}{T}\frac{\lambda_1^2}{\lambda_1+\lambda_2}$.

Proof. Let w, w_r be solutions of the problem defined in (3.1.2), then using $v_r = w + w_r$ in order to eliminate v_r we get

$$\lambda_1\frac{1}{T}\sum_{r=1}^{T}\|w_r\|^2 + \lambda_2\|w\|^2 = \lambda_1\frac{1}{T}\sum_{r=1}^{T}\|v_r - w\|^2 + \lambda_2\|w\|^2\ .$$

Then, using Lemma 3.1.1 in order to eliminate w, we obtain

$$\begin{aligned} &\lambda_1\frac{1}{T}\sum_{r=1}^{T}\|v_r - w\|^2 + \lambda_2\|w\|^2 \\ &= \lambda_1\frac{1}{T}\sum_{r=1}^{T}\|v_r\|^2 - 2\lambda_1\frac{1}{T}\sum_{r=1}^{T}\langle v_r, w\rangle + \lambda_1\|w\|^2 + \lambda_2\|w\|^2 \\ &= \lambda_1\frac{1}{T}\sum_{r=1}^{T}\|v_r\|^2 - 2\lambda_1\frac{1}{T}\sum_{r=1}^{T}\langle v_r, \left(\frac{\lambda_1}{\lambda_1+\lambda_2}\frac{1}{T}\right)\sum_{s=1}^{T}v_s\rangle + (\lambda_1+\lambda_2)\left\|\left(\frac{\lambda_1}{\lambda_1+\lambda_2}\frac{1}{T}\right)\sum_{s=1}^{T}v_s\right\|^2 \\ &= \lambda_1\frac{1}{T}\sum_{r=1}^{T}\|v_r\|^2 - 2\frac{\lambda_1^2}{\lambda_1+\lambda_2}\langle\frac{1}{T}\sum_{r=1}^{T}v_r, \frac{1}{T}\sum_{s=1}^{T}v_s\rangle + (\lambda_1+\lambda_2)\left\|\left(\frac{\lambda_1}{\lambda_1+\lambda_2}\frac{1}{T}\right)\sum_{s=1}^{T}v_s\right\|^2 \\ &= \frac{\lambda_1}{T}\sum_{r=1}^{T}\|v_r\|^2 - \left(2\frac{\lambda_1^2}{\lambda_1+\lambda_2}\right)\left\|\frac{1}{T}\sum_{r=1}^{T}v_r\right\|^2 + \left(\frac{\lambda_1^2}{\lambda_1+\lambda_2}\right)\left\|\frac{1}{T}\sum_{s=1}^{T}v_s\right\|^2 \\ &= \frac{\lambda_1}{T}\sum_{r=1}^{T}\|v_r\|^2 - \left(\frac{\lambda_1^2}{\lambda_1+\lambda_2}\frac{1}{T^2}\right)\left\|\sum_{s=1}^{T}v_s\right\|^2 = A\ . \end{aligned}$$

Also the following regularization term of the objective function that appears in (3.1.6),

$$\rho_1\sum_{r=1}^{T}\|v_r\|^2 + \rho_2\sum_{r=1}^{T}\left\|v_r - \frac{1}{T}\sum_{s=1}^{T}v_s\right\|^2\ ,$$

can be written as follows

$$\begin{aligned} &\rho_1\sum_{r=1}^{T}\|v_r\|^2 + \rho_2\sum_{r=1}^{T}\left\|v_r - \frac{1}{T}\sum_{s=1}^{T}v_s\right\|^2 \\ &= \rho_1\sum_{r=1}^{T}\|v_r\|^2 + \rho_2\sum_{r=1}^{T}\|v_r\|^2 + \rho_2\frac{1}{T}\left\|\sum_{r=1}^{T}v_r\right\|^2 - \rho_2\frac{2}{T}\langle\sum_{r=1}^{T}v_r, \sum_{r=1}^{T}v_r\rangle \\ &= (\rho_1+\rho_2)\sum_{r=1}^{T}\|v_r\|^2 - \rho_2\frac{1}{T}\left\|\sum_{r=1}^{T}v_r\right\|^2 = B\ . \end{aligned}$$

Making A equal to B and solving it is obtained that

$$\begin{cases} \rho_1 + \rho_2 = \frac{\lambda_1}{T} \\ \rho_2 = \frac{\lambda_1^2}{\lambda_1+\lambda_2}\frac{1}{T} \end{cases} \implies \begin{cases} \rho_1 = \frac{1}{T}\frac{\lambda_1\lambda_2}{\lambda_1+\lambda_2} \\ \rho_2 = \frac{1}{T}\frac{\lambda_1^2}{\lambda_1+\lambda_2} \end{cases} ;$$

therefore both formulations of the objective function are equivalent. □

The objective function in the problem defined in (3.1.6), as usual, has an error term and a regularization term. In this case, the regularization term has two parts, the first one,

$$\sum_{r=1}^{T} \|v_r\|^2 ,$$

is the sum of the margins of each individual classifier, while the second one,

$$\sum_{r=1}^{T} \left\| v_r - \frac{1}{T}\sum_{s=1}^{T} v_s \right\|^2 ,$$

can be formally seen as the variance of the weights. That is, since the models for each task r are determined by the weights v_r, this second term measures what we define as the multitask-variance of the models. This means that a small value of the second term implies that the models for each task are very similar to one another, while a large value of this multitask-variance implies differences in the models for different tasks. Knowing the meaning of each term, the parameters ρ_1 and ρ_2 tune the importance given to each of these terms; thus we can associate a particular behaviour to different values of λ_1 and λ_2. If λ_1 is much larger than λ_2 then ρ_2 is larger than ρ_1; hence the multitask-variance of the models will be penalized, resulting in a set of models v_r that are very similar and do not have large margins. However, if λ_1 is much smaller than λ_2, then ρ_2 is smaller than ρ_1; hence the sum of the margins will be penalized, resulting in very different models, each of which with large margins. This confirms what intuitively has been expressed when the multi-task problem was formulated. In conclusion, there are multiple weights v_r generated using different data and we can tune the hyper-parameters to penalize the variance or the margin of each model. Therefore, there is a trade-off between large margins and large multitask-variance when obtaining the models.

3.1.2 Formulation as a Single-Task Problem

After the analysis of the previous section, in order to make a practical use of this multi-task problem definition it is necessary to develop a method that solves the problem of optimization defined in (3.1.2). Note that the primal defined in the formulation (3.1.2) is no longer used. The first attempt for solving (3.1.2) is trying to adapt the classical approach to SVM to this problem; however this idea fails since it is necessary to estimate T weight vectors instead of a single one. On the other side, it is possible to change the notation and write the multi-task problem as a classical single-task one, namely

$$\begin{aligned} \underset{w,\xi}{\arg\min} \quad & J(w,\xi) = C\sum_{i=1}^{N}\xi_i + \frac{1}{2}||w||^2 \\ \text{s.t.} \quad & y_i w \cdot x_i \geq p_i - \xi_i,\ i \in \{1,\dots,N\}, \\ & \xi_i \geq 0,\ i \in \{1,\dots,N\} . \end{aligned}$$

In order to do so the mapping $\Phi(x, r)$ is defined as follows

$$\Phi(x,r) = \left(\overbrace{\frac{x}{\sqrt{\mu}}}^{0}, \overbrace{\mathbf{0}}^{1}, \ldots, \overbrace{\mathbf{0}}^{r-1}, \overbrace{x}^{r}, \overbrace{\mathbf{0}}^{r+1}, \ldots, \overbrace{\mathbf{0}}^{T} \right) ,$$

where $\mathbf{0} \in \mathbb{R}^d$ and

$$\mu = \frac{T\lambda_2}{\lambda_1} . \tag{3.1.7}$$

If we assume that the original data x comes from $\mathbb{R}^d$, then it is easy to observe that $\Phi(x, r)$ is a vector of dimension $d \times (T+1)$, that is $\Phi(x, r) \in \mathbb{R}^{d\times(T+1)}$. After this, the next step is to define a weight vector $\boldsymbol{w}^\mu$ that matches the mapping $\Phi(x, r)$; thus, the following definition is given

$$\boldsymbol{w}^\mu = (\overbrace{\sqrt{\mu}w}^{0}, \overbrace{w_1}^{1}, \overbrace{w_2}^{2}, \ldots, \overbrace{w_r}^{r}, \ldots, \overbrace{w_T}^{T}) .$$

With this definition, the weight vector $\boldsymbol{w}^\mu$ has the same dimension as $\Phi(x, r)$; hence, we can write the following dot product

$$\begin{aligned}
&\boldsymbol{w}^\mu \cdot \Phi(x,r) \\
&= (\overbrace{\sqrt{\mu}w}^{0}, \overbrace{w_1}^{1}, \overbrace{w_2}^{2}, \ldots, \overbrace{w_r}^{r}, \ldots, \overbrace{w_T}^{T}) \cdot \left(\overbrace{\frac{x}{\sqrt{\mu}}}^{0}, \overbrace{\mathbf{0}}^{1}, \ldots, \overbrace{\mathbf{0}}^{r-1}, \overbrace{x}^{r}, \overbrace{\mathbf{0}}^{r+1}, \ldots, \overbrace{\mathbf{0}}^{T} \right) \\
&= wx + w_r x = v_r x .
\end{aligned}$$

The result of the dot product $\boldsymbol{w}^\mu \cdot \Phi(x, r)$ is $wx + w_r x$, which is just the product that appears in the restrictions for the problem defined in (3.1.2). Moreover, the square norm of $\boldsymbol{w}^\mu$ is

$$\begin{aligned}
\|\boldsymbol{w}^\mu\|^2 &= (\overbrace{\sqrt{\mu}w}^{0}, \overbrace{w_1}^{1}, \overbrace{w_2}^{2}, \ldots, \overbrace{w_r}^{r}, \ldots, \overbrace{w_T}^{T}) \cdot (\overbrace{\sqrt{\mu}w}^{0}, \overbrace{w_1}^{1}, \overbrace{w_2}^{2}, \ldots, \overbrace{w_r}^{r}, \ldots, \overbrace{w_T}^{T}) \\
&= \mu \|w\|^2 + \sum_{r=1}^{T} \|w_r\|^2 = \frac{T\lambda_2}{\lambda_1} \|w\|^2 + \sum_{r=1}^{T} \|w_r\|^2 .
\end{aligned}$$

It is easily noticeable that this is proportional to the regularization term in (3.1.2):

$$\frac{\lambda_1}{T} \|\boldsymbol{w}^\mu\|^2 = \lambda_2 \|w_0\|^2 + \frac{\lambda_1}{T} \sum_{t=1}^{T} \|v_t\|^2.$$

Using these definitions and results, the multi-task primal problem previously defined in (3.1.2) can be rewritten as

$$\begin{aligned}
\underset{\boldsymbol{w}^\mu,\xi}{\arg\min} \quad & J(\boldsymbol{w}^\mu, \xi) = \sum_{i=1}^{N} \xi_i + \frac{\lambda_1}{T} \|\boldsymbol{w}^\mu\|^2 \\
\text{s.t.} \quad & y_i \boldsymbol{w}^\mu \cdot \Phi(x_i, t_i) \geq p_i - \xi_i, \; i \in \{1, \ldots, N\}, \\
& \xi_i \geq 0, \; i \in \{1, \ldots, N\} ,
\end{aligned} \tag{3.1.8}$$

where $N = \sum_{r=1}^{T} m_r$. Multiplying the objective function $J(\boldsymbol{w}^\mu, \xi)$ times $C = \frac{T}{2\lambda_1}$ we get

$$\begin{aligned} \underset{\boldsymbol{w}^\mu,\xi}{\arg\min} \quad & J(\boldsymbol{w}^\mu, \xi) = C \sum_{i=1}^{N} \xi_i + \frac{1}{2} \left\| \boldsymbol{w}^\mu \right\|^2 \\ \text{s.t.} \quad & y_i \boldsymbol{w}^\mu \cdot \Phi(x_i, t_i) \geq p_i - \xi_i, \ i \in \{1, \ldots, N\}, \\ & \xi_i \geq 0, \ i \in \{1, \ldots, N\} \ . \end{aligned} \tag{3.1.9}$$

Therefore, the multi-task problem can be rewritten as a single-task problem, in which we can apply standard methods for solving SVM. However is important to highlight that in the dual problem of the SVM we have a matrix Q, in which typically $Q_{ij} = x_i \cdot x_j$ (assuming a linear kernel). In this particular case of multi-task learning the matrix $\hat{Q}$ is defined as

$$\hat{Q}_{ij} = \Phi(x_i, r(i)) \cdot \Phi(x_j, r(j)) \ ,$$

where $r(i)$ is the task to which x_i corresponds to. Therefore we can write $\hat{Q}$, the matrix corresponding to the quadratic term in the dual problem, as follows,

$$\begin{aligned} \hat{Q}_{ij} &= \Phi(x_i, r(i)) \cdot \Phi(x_j, r(j)) \\ &= \left(\frac{\overbrace{x}^{0}}{\sqrt{\mu}}, \overbrace{\mathbf{0}}^{1}, \ldots, \overbrace{\mathbf{0}}^{r-1}, \overbrace{x}^{r}, \overbrace{\mathbf{0}}^{r+1}, \ldots, \overbrace{\mathbf{0}}^{T} \right) \cdot \left(\frac{\overbrace{x}^{0}}{\sqrt{\mu}}, \overbrace{\mathbf{0}}^{1}, \ldots, \overbrace{\mathbf{0}}^{r-1}, \overbrace{x}^{r}, \overbrace{\mathbf{0}}^{r+1}, \ldots, \overbrace{\mathbf{0}}^{T} \right) \\ &= \left(\frac{1}{\mu} + \delta_{r(i)r(j)} \right) x_i \cdot x_j \ . \end{aligned}$$

Then, if Q is the matrix defined so that in the position Q_{ij} has the element $x_i \cdot x_j$, then

$$\hat{Q} = \frac{1}{\mu} Q + K \ ,$$

where K is a block diagonal matrix with the following form

$$K = \begin{pmatrix} K_1 & \mathbf{0}^{m_1 \times m_2} & \mathbf{0}^{m_1 \times m_3} & \ldots & \mathbf{0}^{m_1 \times m_T} \\ \mathbf{0}^{m_2 \times m_1} & K_2 & \mathbf{0}^{m_2 \times m_1} & \ldots & \mathbf{0}^{m_2 \times m_T} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{0}^{m_T \times m_1} & \mathbf{0}^{m_T \times m_2} & \mathbf{0}^{m_T \times m_3} & \ldots & K_T \end{pmatrix} \ ,$$

with

$$K_r = \begin{pmatrix} x_1^r \cdot x_1^r & x_1^r \cdot x_2^r & \ldots & x_1^r \cdot x_{m_r}^r \\ x_2^r \cdot x_1^r & x_2^r \cdot x_2^r & \ldots & x_2^r \cdot x_{m_r}^r \\ \vdots & \vdots & \ddots & \vdots \\ x_{m_r}^r \cdot x_1^r & x_{m_r}^r \cdot x_2^r & \ldots & x_{m_r}^r \cdot x_{m_r}^r \end{pmatrix} \ ,$$

and $\mathbf{0}^{m_r \times m_s}$ is the zero matrix with size $m_r \times m_s$. Given this observation the dual can be written as follows

$$\begin{aligned} \underset{\alpha}{\arg\min} \quad & \Theta(\alpha) = \frac{1}{2} \alpha^t \left(\frac{1}{\mu} Q + K \right) \alpha - p\alpha \\ \text{s.t.} \quad & 0 \leq \alpha_i^r \leq C, \ i = 1, \ldots, m_r; r = 1, \ldots, T \ . \end{aligned} \tag{3.1.10}$$

Therefore, once we have written the multi-task problem with a single-task formulation we can use algorithms like SMO to solve it. Nevertheless, it is important to note that the bias

has been omitted since it is assumed to be null, and thus there is no equality constraint; therefore, linear methods such as the one presented in [16] could be used. However, the assumption of a null bias is not suitable for every problem; although there are methods like centering the data around zero and then assuming that the bias is not needed, it is not an optimal practice. This fact is one of the main limitations of the *Regularized Multitask Learning* algorithm. Another big limitation of this method is the assumption of a linear kernel; we can observe that the kernel defined in (3.1.10) is just a linear kernel shrunk by μ and with the blocks on the diagonal being reinforced. These two limitations make the rMTL method not very attractive except for some very particular cases where the bias is zero and the assumption of linearity is valid. It is therefore necessary to extend the idea to a more general framework. In the next subsection a generalization of the multi-task learning problem defined in (3.1.2) will be presented, as well as a solution for both the bias and the linear kernel problems.

3.2 Multiple Kernel Support Vector Machines for Multitask Learning

In our initial approach, the one corresponding to Regularized Multitask Learning, we assumed linearity and thus data were not mapped into another Hilbert space; instead the original feature space is used. In particular, the space where the data belong was the same for both the common part w and the specific part w_r of each model. However, in the work carried out in [2] a different approximation is made. The main idea is to split the data simultaneously into two Hilbert spaces:

- **Decision Space**: This is where the data are mapped into for the common part of the model. It is represented as $\phi(x_i^r)$.
- **Correction Space**: This is where the data are mapped into for the specific part of the model. It can be different for each task r and it is represented as $\phi_r(x_i^r)$.

These transformations can detect non-linear relations that in rMTL, as it is defined in [1], were omitted. Moreover, in the previous approach the bias was eliminated which resulted in models that are not expressive enough to adapt to data that are not centered around zero. In this new approach a bias term is added in order to get more expressive models. Therefore, the multi-task primal problem following this definitions is the following one,

$$\begin{aligned} \underset{w,w_r,\xi}{\arg\min} \quad & J(w, w_r, \xi) = C \sum_{r=1}^{T} \sum_{i=1}^{m} \xi_i^r + \frac{1}{2} \sum_{r=1}^{T} \|w_r\|^2 + \frac{\mu}{2} \|w\|^2 \\ \text{s.t.} \quad & y_i^r (w \cdot \phi(x_i^r) + b + w_r \cdot \phi_r(x_i^r) + b_r) \geq p_i^r - \xi_i^r, \; i = 1, \ldots, m_r; \; r = 1, \ldots, T, \\ & \xi_i^r \geq 0, \; i = 1, \ldots, m_r; \; r = 1, \ldots, T \; . \end{aligned} \tag{3.2.1}$$

It is relevant to notice two aspects. In first place the introduction of the bias term is made in this approach, which solves one of the main flaws of Regularized Multitask Learning. Moreover it is important to notice that the bias term is also split in a common part b and a specific part b_r. In second place, it can be seen that, when the bias term is removed, this approach is just a generalization of rMTL. When choosing all transformations ϕ_r and ϕ as the identity, it can be seen that when defining C and μ such that $\mu = \frac{T\lambda_2}{\lambda_1}$ and $C = \frac{T}{\lambda_1}$, then the objective function in (3.2.1) is C times the one in (3.1.2). Given this observation the knowledge about the influence of λ_1 and λ_2 in (3.1.2) can be extended to the influence of μ in (3.2.1). That is, choosing μ small penalizes the specific part, resulting in very similar

models for each task. On the other hand, choosing μ large results in different models for different tasks. Finally, to put this formulation in relation with the single-task model, C plays the same role in both single and multi-task models; it tunes the importance between margins and errors; the additional parameter μ balances the importance between margins in each of the different mapping spaces and the similarity among the different models.

Since the bias term is being taken into account, the formulation given in the previous section in (3.1.10), where the problem is expressed as a single-task problem, is not complete. The main reason behind this is that it is not possible to express every bias term of individual models, namely $b + b_r$, as an unique bias for the single-task formulation used in (3.1.9). As a consequence of this issue, another solution has to be presented and it makes use of the Lagrangian Theory. In order to do so the dual problem and the KKT conditions of the multiple kernel multi-task learning problem will be developed. In first place, the Lagrangian corresponding to (3.2.1) is the following one,

$$\begin{aligned}\mathcal{L}(w, w_r, b, b_r, \alpha, \beta) = C \sum_{r=1}^{T} \sum_{i=1}^{m} \xi_i^r + \frac{1}{2} \sum_{r=1}^{T} \|w_r\|^2 + \frac{\mu}{2} \|w\|^2 \\ - \sum_{r=1}^{T} \sum_{i=1}^{n_r} \alpha_i^r [y_i^r (w \cdot \phi(x_i^r) + b + w_r \cdot \phi_r(x_i^r) + b_r) - p_i^r + \xi_i^r] \\ - \sum_{r=1}^{T} \sum_{i=1}^{n_r} \beta_i^r \xi_i^r \ .\end{aligned} \tag{3.2.2}$$

In order to write the dual problem, the primal variables are eliminated: computing the gradient with respect to the variables of the primal problem and solving for zero we get the following results:

$$\frac{\partial \mathcal{L}}{\partial w} = 0 \implies w^* = \frac{1}{\mu} \sum_{r=1}^{T} \sum_{i=1}^{n_r} \alpha_i^r y_i^r \phi(x_i^r) \ , \tag{3.2.3}$$

$$\frac{\partial \mathcal{L}}{\partial w_r} = 0 \implies w_r^* = \sum_{i=1}^{n_r} \alpha_i^r y_i^r \phi_r(x_i^r) \ , \tag{3.2.4}$$

$$\frac{\partial \mathcal{L}}{\partial b} = 0 \implies \sum_{r=1}^{T} \sum_{i=1}^{n_r} \alpha_i^r y_i^r = 0 \ , \tag{3.2.5}$$

$$\frac{\partial \mathcal{L}}{\partial b_r} = 0 \implies \sum_{i=1}^{n_r} \alpha_i^r y_i^r = 0 \ , \tag{3.2.6}$$

$$\frac{\partial \mathcal{L}}{\partial \xi_i^r} = 0 \implies C - \alpha_i^r - \beta_i^r = 0 \ . \tag{3.2.7}$$

Before writing the dual problem, some observations can be made about the previous results. Note that equations (3.2.5) and (3.2.6) yield a similar result and it is easy to see that the first one is a consequence of the second one. That means that, in contrast to the single-task problem, there are T different equality constraints corresponding to (3.2.6) and the condition (3.2.5) can be eliminated. The dual problem is defined as

$$\arg\max_{0 \leq \alpha, \beta \leq C} \Theta(\alpha, \beta) = \arg\max_{\alpha, \beta} \min_{w, w_r, b, b_r} \mathcal{L}(w, w_r, b, b_r, \alpha, \beta) \ .$$

Using the previous results we can write the dual objective function of the multi-task prob-

lem in the following way

$$
\begin{aligned}
\Theta(\alpha) = \mathcal{L}(w^*, w_r^*, b^*, b_r^*, \alpha, \beta) &= \frac{1}{2}\sum_{r=1}^{T}\|w_r^*\|^2 + \frac{\mu}{2}\|w^*\|^2 - \\
&\sum_{r=1}^{T}\sum_{i=1}^{n_r}\alpha_i^r[y_i^r(w^*\cdot\phi(x_i^r) + w_r^*\cdot\phi_r(x_i^r)) - p_i^r] \\
&= \frac{1}{2}\sum_{r=1}^{T}\left\|\sum_{i=1}^{n_r}\alpha_i^r y_i^r \phi_r(x_i^r)\right\|^2 + \frac{\mu}{2}\left\|\frac{1}{\mu}\sum_{r=1}^{T}\sum_{i=1}^{n_r}\alpha_i^r y_i^r \phi(x_i^r)\right\|^2 - \\
&\sum_{r=1}^{T}\sum_{i=1}^{n_r}\alpha_i^r\left\{y_i^r\left(\left(\frac{1}{\mu}\sum_{s=1}^{T}\sum_{j=1}^{n_s}\alpha_j^s y_j^s\phi(x_j^s)\right)\cdot\phi(x_i^r) + \left(\sum_{j=1}^{n_r}\alpha_i^r y_j^r\phi_r(x_j^r)\right)\cdot\phi_r(x_i^r)\right) - p_i^r\right\} \\
&= \frac{1}{2}\sum_{r=1}^{T}\langle\sum_{i=1}^{n_r}\alpha_i^r y_i^r\phi_r(x_i^r), \sum_{i=1}^{n_r}\alpha_i^r y_i^r\phi_r(x_i^r)\rangle + \frac{1}{2}\frac{1}{\mu}\langle\sum_{r=1}^{T}\sum_{i=1}^{n_r}\alpha_i^r y_i^r\phi(x_i^r), \sum_{r=1}^{T}\sum_{i=1}^{n_r}\alpha_i^r y_i^r\phi(x_i^r)\rangle - \\
&\frac{1}{\mu}\sum_{r=1}^{T}\sum_{i=1}^{n_r}\left(\sum_{s=1}^{T}\sum_{j=1}^{n_s}\alpha_j^s y_j^s\phi(x_j^s)\right)\cdot\alpha_i^r y_i^r\phi(x_i^r) - \sum_{r=1}^{T}\sum_{i=1}^{n_r}\sum_{j=1}^{n_r}\alpha_i^r y_j^r\phi_r(x_j^r)\cdot\alpha_i^r y_i^r\phi_r(x_i^r) + \\
&\sum_{r=1}^{T}\sum_{i=1}^{n_r}\alpha_i^r p_i^r \\
&= -\frac{1}{2\mu}\sum_{r=1}^{T}\sum_{i=1}^{n_r}\left(\sum_{s=1}^{T}\sum_{j=1}^{n_s}\alpha_j^s y_j^s\phi(x_j^s)\right)\cdot\alpha_i^r y_i^r\phi(x_i^r) - \frac{1}{2}\sum_{r=1}^{T}\sum_{i=1}^{n_r}\sum_{j=1}^{n_r}\alpha_i^r y_j^r\phi_r(x_j^r)\cdot\alpha_i^r y_i^r\phi_r(x_i^r) + \\
&\sum_{r=1}^{T}\sum_{i=1}^{n_r}\alpha_i^r p_i^r
\end{aligned}
$$

Therefore the dual problem is

$$
\begin{aligned}
\arg\min_{\alpha} \quad \Theta(\alpha) &= \frac{1}{2}\frac{1}{\mu}\sum_{r=1}^{T}\sum_{i=1}^{n_r}\sum_{s=1}^{T}\sum_{j=1}^{n_s}\alpha_i^r\alpha_j^s y_i^r y_j^s\phi(x_i^r)\cdot\phi(x_j^s) + \\
&\frac{1}{2}\sum_{r=1}^{T}\sum_{i=1}^{n_r}\sum_{j=1}^{n_r}\alpha_i^r\alpha_j^r y_i^r y_j^r\phi_r(x_i^r)\cdot\phi_r(x_j^r) - \sum_{r=1}^{T}\sum_{i=1}^{n_r}\alpha_i^r p_i^r \\
\text{s.t.} \quad & 0 \le \alpha_i^r \le C,\ i = 1,\dots,m_r; r = 1,\dots,T \text{ (box constraints)}, \\
& \sum_{i=1}^{n_r}\alpha_i^r y_i^r = 0,\ r = 1,\dots,T \text{ (equality constraints)}.
\end{aligned}
\tag{3.2.8}
$$

It is also helpful to write the dual problem in vector notation, that is

$$
\begin{aligned}
\arg\min_{\alpha} \quad & \Theta(\alpha) = \frac{1}{2}\alpha^t\hat{Q}\alpha - p\alpha \\
\text{s.t.} \quad & 0 \le \alpha_i^r \le C,\ i = 1,\dots,m_r; r = 1,\dots,T \text{ (box constraints)}, \\
& \sum_{i=1}^{n_r}\alpha_i^r y_i^r = 0,\ r = 1,\dots,T \text{ (equality constraints)},
\end{aligned}
\tag{3.2.9}
$$

where Q is the matrix in which the position Q_{ij} contains $k(x_i, x_j) = \phi(x_i) \cdot \phi(x_j)$ and

$$K = \begin{pmatrix} K_1 & \mathbf{0}^{m_1 \times m_2} & \mathbf{0}^{m_1 \times m_3} & \dots & \mathbf{0}^{m_1 \times m_T} \\ \mathbf{0}^{m_2 \times m_1} & K_2 & \mathbf{0}^{m_2 \times m_1} & \dots & \mathbf{0}^{m_2 \times m_T} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{0}^{m_T \times m_1} & \mathbf{0}^{m_T \times m_2} & \mathbf{0}^{m_T \times m_3} & \dots & K_T \end{pmatrix} ,$$

with

$$K_r = \begin{pmatrix} k_r(x_1^r, x_1^r) & k_r(x_1^r, x_2^r) & \dots & k_r(x_1^r, x_{m_r}^r) \\ k_r(x_2^r, x_1^r) & k_r(x_2^r, x_2^r) & \dots & k_r(x_2^r, x_{m_r}^r) \\ \vdots & \vdots & \ddots & \vdots \\ k_r(x_{m_r}^r, x_1^r) & k_r(x_{m_r}^r, x_2^r) & \dots & k_r(x_{m_r}^r, x_{m_r}^r) \end{pmatrix} ,$$

being $k_r(x_i^r, x_j^r) = \phi_r(x_i^r) \cdot \phi_r(x_j^r)$. Given this definitions it can be seen that this is a generalization of rMTL, where the $\hat{Q}$ matrix has multiple kernels. Apart from this, the main difference between this approach and the previous one is the introduction of the bias, which results in T equality constraints that have to be fulfilled when solving the problem. A method that overcomes this difficulty is presented in the next section.

3.3 GSMO Algorithm

The multi-task problem defined in the previous section presents certain challenges with respect to the single-task problem. The definition of the primal problem involves a common part w and a specific part w_r for every model, and it seems that this fact implies that the single and multi-task problems are independent ones. However, when looking at the dual problems of both single and multi-task approaches it can be seen that there are mainly two differences between them. That is, a different matrix in the quadratic term of the dual problem, and multiple equality constraints that have to be fulfilled. In order to overcome this difficulties the SMO algorithm is adapted in [2] receiving the name of *Generalized SMO* (GSMO). This sections gives a detailed explanation of the adapted algorithm and it also presents a computational cost discussion.

3.3.1 Update Step

The GSMO algorithm generalizes the SMO algorithm in order to match the multi-task problem requirements. Before going into detail it is important to define our goal, which is to find the solutions w^*, w_r^* to the multi-task primal problem (3.2.1). In order to do so, and using optimization theory, we solve the dual, which can be written as

$$\Theta(\alpha) = \Theta(\alpha^1, \dots, \alpha^T) = \Theta(\alpha_1^1, \dots, \alpha_{m_1}^1, \dots, \alpha_1^T, \dots, \alpha_{m_T}^T) ,$$

where α_i^r is the i-th dual variable of task r, α is the vector with every dual variable and α^r is the vector with all the dual variables of task r. It is important to notice that the primal variables w^* and w_r^* depend only on α; hence, using (3.2.3) and (3.2.4), we can recover w^* and w_r^* after finding an optimal α^*. This can be done by minimizing $\Theta(\alpha)$, while keeping α feasible, until the KKT conditions are met, which implies optimality as we have seen in Chapter 2.

To do so, in first place it is important to notice that there are T different equality constraints as shown in (3.2.6), one for the variables corresponding to each task; thus it is

necessary to update two variables of the same task r simultaneously. In this algorithm, at each iteration we first select the task r whose variables we want to update, and then we update only those variables. For this reason, we will first see how to update the variables once the task r has been chosen, and later it will be shown how to select such task. Let U_r and L_r be the indices of the variables chosen to be updated corresponding to task r; that is, the dual variables $\alpha^r_{U_r}, \alpha^r_{L_r}$ will be updated; however, aiming to keep the notation simple we will denote them as $\alpha_{U_r}, \alpha_{L_r}$. Then, as in SMO, the update is the following

$$\begin{aligned}\alpha_{U_r}(\tau+1) &= \alpha_{U_r}(\tau) + \delta_{U_r}(\tau)\ ,\\ \alpha_{L_r}(\tau+1) &= \alpha_{L_r}(\tau) + \delta_{L_r}(\tau)\ .\end{aligned}$$

Using the equality constraint

$$\sum_{i=0}^{m_r} \alpha^r_i y^r_i = 0\ ,$$

and operating like in the SMO proof we obtain.

$$\delta_{U_r}(\tau) y_{U_r} + \delta_{L_r}(\tau) y_{L_r} = 0\ . \tag{3.3.1}$$

Using this equation it is possible to write the algorithm in terms of just δ_{L_r}, just as in the traditional SMO algorithm. The next step is to express the difference $\Theta^\tau - \Theta^{\tau+1}$ as function of δ_{L_r}, that is

$$\Theta^\tau - \Theta^{\tau+1} = \psi(\delta_{L_r})\ .$$

Imitating the single-task proof SMO and denoting $\hat{Q}$ as

$$\hat{Q} = \left(\frac{1}{\mu}Q + K\right)\ ,$$

we have

$$\begin{aligned}\Theta^\tau - \Theta^{\tau+1} &= \frac{1}{2}\alpha(\tau)^t\hat{Q}\alpha(\tau) - p\alpha(\tau)\\ &\quad - \left(\frac{1}{2}\alpha(\tau+1)^t\hat{Q}\alpha(\tau+1) - p\alpha(\tau+1)\right)\\ &= \frac{1}{2}\alpha(\tau)^t\hat{Q}\alpha(\tau) - p\alpha(\tau)\\ &\quad - \left(\frac{1}{2}(\alpha(\tau)+\delta(\tau))^t\hat{Q}(\alpha(\tau)+\delta(\tau)) - p(\alpha(\tau)+\delta(\tau))\right)\\ &= -\frac{1}{2}\left(\delta(\tau)^t\hat{Q}\delta(\tau) + \alpha(\tau)^t\hat{Q}\delta(\tau) + \delta(\tau)^t\hat{Q}\alpha(\tau)\right) + p\delta(\tau)\\ &= -\frac{1}{2}\delta(\tau)^t\hat{Q}\delta(\tau) - \left(\delta(\tau)^t\hat{Q}\alpha(\tau) - p\delta(\tau)\right)\ .\end{aligned} \tag{3.3.2}$$

As in the SMO proof we omit the iteration time τ, resulting in the following equation

$$\Theta^\tau - \Theta^{\tau+1} = -\frac{1}{2}\delta^t\hat{Q}\delta - \left(\delta^t\hat{Q}\alpha - p\delta\right) = -\frac{1}{2}A - B\ . \tag{3.3.3}$$

where

$$\alpha = \left(\alpha^1_1, \ldots, \alpha^1_{m_1}, \ldots, \alpha^T_1, , \ldots, \alpha^T_{m_T}\right)^t$$

and

$$\delta = (0, \ldots, 0, \delta_{L_r}, 0, \ldots, 0, \delta_{U_r}, 0 \ldots, 0)^t\ ,$$

where only the positions L_r and U_r are not null. Given these clarifications, (3.3.3) can be further expanded. The term denoted as A can be written as follows

$$\begin{aligned}
A &= \delta^t \hat{Q} \delta \\
&= \delta_{U_r}^2 \hat{Q}_{U_r U_r} + 2\delta_{U_r}\delta_{L_r}\hat{Q}_{U_r L_r} + \delta_{L_r}^2 \hat{Q}_{L_r L_r} \\
&= \frac{1}{\mu}\left(\delta_{U_r}^2 Q_{U_r U_r} + 2\delta_{U_r}\delta_{L_r} Q_{U_r L_r} + \delta_{L_r}^2 Q_{L_r L_r}\right) + \\
&\quad \left(\delta_{U_r}^2 K_{U_r U_r} + 2\delta_{U_r}\delta_{L_r} K_{U_r L_r} + \delta_{L_r}^2 K_{L_r L_r}\right) \\
&= \frac{1}{\mu}\left(\delta_{L_r}^2 \phi(x_{U_r})\cdot\phi(x_{U_r}) + 2(-y_U y_L \delta_{L_r})\delta_{L_r} y_U y_L \phi(x_{U_r})\cdot\phi(x_{L_r}) + \delta_{L_r}^2 \phi(x_{L_r})\cdot\phi(x_{L_r})\right) + \\
&\quad \delta_{L_r}^2 \phi_r(x_{U_r})\cdot\phi_r(x_{U_r}) + \\
&\quad 2(-y_U y_L \delta_{L_r})\delta_{L_r} y_U y_L \phi_r(x_{U_r})\cdot\phi_r(x_{L_r}) + \delta_{L_r}^2 \phi_r(x_{L_r})\cdot\phi_r(x_{L_r})) \\
&= \frac{1}{\mu}\left(\delta_{L_r}^2 \left\|\phi(x_{U_r})\right\|^2 - 2\delta_{L_r}^2 \phi(x_{U_r})\cdot\phi(x_{L_r}) + \delta_{L_r}^2 \left\|\phi(x_{L_r})\right\|^2\right) + \\
&\quad \left(\delta_{L_r}^2 \left\|\phi_r(x_{U_r})\right\|^2 - 2\delta_{L_r}^2 \phi_r(x_{U_r})\cdot\phi_r(x_{L_r}) + \delta_{L_r}^2 \left\|\phi_r(x_{L_r})\right\|^2\right) \\
&= \delta_{L_r}^2 \left(\frac{1}{\mu}\left\|\phi(x_{U_r}) - \phi(x_{L_r})\right\|^2 + \left\|\phi_r(x_{U_r}) - \phi_r(x_{L_r}\right\|^2)\right) .
\end{aligned}$$

We can notice that the term corresponding to the norm of the difference is different in GSMO because we use $\hat{Q}$ instead of Q. Besides, the term denoted as B can be also expanded in the following way,

$$\begin{aligned}
B = \delta\hat{Q}\alpha - p\delta &= \delta_{L_r}\left(\left(\hat{Q}\alpha\right)_{L_r} - p_{L_r}\right) + \delta_{U_r}\left(\left(\hat{Q}\alpha\right)_{U_r} - p_{U_r}\right) \\
&= \delta_{L_r}\left(\sum_{i=1}^{N}\hat{Q}_{L_r i}\alpha_i - p_{L_r}\right) + \delta_{U_r}\left(\sum_{i=1}^{N}\hat{Q}_{U_r i}\alpha_i - p_{U_r}\right) \\
&= \delta_{L_r}\left(\sum_{i=1}^{N}\hat{Q}_{L_r i}\alpha_i - p_{L_r}\right) - y_{U_r}y_{L_r}\delta_{L_r}\left(\sum_{i=1}^{N}\hat{Q}_{U_r i}\alpha_i - p_{U_r}\right) \\
&= y_{L_r}\delta_{L_r}\left(y_{L_r}\left(\sum_{i=1}^{N}\hat{Q}_{L_r i}\alpha_i - p_{L_r}\right) - y_{U_r}\left(\sum_{i=1}^{N}\hat{Q}_{U_r i}\alpha_i - p_{U_r}\right)\right) ,
\end{aligned}$$

where N is the total number of examples of all tasks; that is

$$N = \sum_{r=1}^{T} m_r .$$

Moreover, following the SMO scheme, the gradient of the dual objective function is the following

$$\nabla\Theta(\alpha) = \hat{Q} - p ; \tag{3.3.4}$$

therefore the position j of the gradient is

$$\nabla\Theta(\alpha)_j = \hat{Q}_j - p_j = \left(\sum_{i=1}^{N}\hat{Q}_{ji}\alpha_i - p_j\right) .$$

Using this, the following is true

$$B = y_{L_r}\delta_{L_r}\left(y_{L_r}(\nabla\Theta)_{L_r} - y_{U_r}(\nabla\Theta)_{U_r}\right) .$$

Finally, using the results for A and B, we obtain the following result

$$\begin{aligned}\Theta^{\tau} - \Theta^{\tau+1} &= -y_{L_r}\delta_{L_r}\left(y_{L_r}(\nabla\Theta)_{L_r} - y_{U_r}(\nabla\Theta)_{U_r}\right) - \\ &\qquad \frac{1}{2}\delta_{L_r}^2\left(\frac{1}{\mu}\left\|\phi(x_{U_r}) - \phi(x_{L_r})\right\|^2 + \left\|\phi_r(x_{U_r}) - \phi_r(x_{L_r}\right\|^2)\right) \\ &= -\psi(\delta_{L_r})\ .\end{aligned} \tag{3.3.5}$$

The goal, as in SMO, is to minimize $\psi(\delta_{L_r})$, by solving $\psi'(\delta_{L_r}^*) = 0$, where

$$\begin{aligned}\psi'(\delta_{L_r}) = y_{L_r}&\left(y_{L_r}(\nabla\Theta)_{L_r} - y_{U_r}(\nabla\Theta)_{U_r}\right) + \\ &\delta_{L_r}\left(\frac{1}{\mu}\left\|\phi(x_{U_r}) - \phi(x_{L_r})\right\|^2 + \left\|\phi_r(x_{U_r}) - \phi_r(x_{L_r}\right\|^2)\right)\ .\end{aligned}$$

The result for GSMO is the following

$$\delta_{L_R}^* = \frac{y_{L_r}\left(y_{U_r}(\nabla\Theta)_{U_r} - y_{L_r}(\nabla\Theta)_{L_r}\right)}{\frac{1}{\mu}\left\|\phi(x_{U_r}) - \phi(x_{L_r})\right\|^2 + \left\|\phi_r(x_{U_r}) - \phi_r(x_{L_r}\right\|^2}\ . \tag{3.3.6}$$

We can define then $\bar{\lambda}$ as

$$\bar{\lambda} = \frac{\left(y_{U_r}(\nabla\Theta)_{U_r} - y_{L_r}(\nabla\Theta)_{L_r}\right)}{\frac{1}{\mu}\left\|\phi(x_{U_r}) - \phi(x_{L_r})\right\|^2 + \left\|\phi_r(x_{U_r}) - \phi_r(x_{L_r}\right\|^2}\ .$$

We will see later that $\bar{\lambda}$ is used for the selection of the task r to update. Following the update step, the update of the dual variables α_{U_r} and α_{L_r} would be the following

$$\begin{aligned}\alpha_{L_r}^{\tau+1} &= \alpha_{L_r}^{\tau} + y_{L_r}\bar{\lambda} \\ \alpha_{U_r}^{\tau+1} &= \alpha_{U_r}^{\tau} - y_{U_r}\bar{\lambda}\ ;\end{aligned}$$

As in SMO, we need to clip the update in the following way,

$$\begin{aligned}&\lambda = \min\left(C - \alpha_{L_r}^{\tau}, \bar{\lambda}\right) \text{ if } y_{L_r} = 1\ , \\ &\lambda = \min\left(\alpha_{L_r}^{\tau}, \bar{\lambda}\right) \text{ if } y_{L_r} = \text{-}1\ , \\ &\lambda = \min\left(\alpha_{U_r}^{\tau}, \bar{\lambda}\right) \text{ if } y_{U_r} = 1\ , \\ &\lambda = \min\left(C - \alpha_{U_r}^{\tau}, \bar{\lambda}\right) \text{ if } y_{U_r} = \text{-}1\ .\end{aligned}$$

Then, we use λ to update the dual variables

$$\begin{aligned}\alpha_{L_r}^{\tau+1} &= \alpha_{L_r}^{\tau} + y_{L_r}\lambda\ , \\ \alpha_{U_r}^{\tau+1} &= \alpha_{U_r}^{\tau} - y_{U_r}\lambda\ .\end{aligned}$$

As in SMO, it can be seen that even with the clipping, the update makes Θ smaller. To check just one case, set $y_{L_r} = -1$ and $\alpha_{L_r}^{\tau} < \bar{\lambda}$, then $\delta_{L_r}^{\tau} = y_{L_r}\alpha_{L_r}^{\tau}$, therefore

$$\begin{aligned}\Theta^{\tau} - \Theta^{\tau+1} &= \psi(y_{L_r}\alpha_{L_r}^{\tau}) = \alpha_{L_r}^{\tau}[y_{U_r}(\nabla\Theta)_{U_r} - y_{L_r}(\nabla\Theta)_{L_r}] \\ &\qquad - \frac{1}{2}\left(\alpha_{L_r}^{\tau}\right)^2\left(\frac{1}{\mu}\left\|\phi(x_{U_r}) - \phi(x_{L_r})\right\|^2 + \left\|\phi_r(x_{U_r}) - \phi_r(x_{L_r}\right\|^2\right) \\ &\geq \alpha_{L_r}^{\tau}[y_{U_r}(\nabla\Theta)_{U_r} - y_{L_r}(\nabla\Theta)_{L_r}] > 0\ .\end{aligned}$$

The last inequality is yet to be proven. Since we are not exactly in the same situation that we had in SMO, it is necessary to review the selection step that we made for a multi-task problem.

3.3.2 Selection Step

Before presenting the heuristic methods that are used in the selection step it is important to know the KKT conditions of this multi-task problem, which are

$$\begin{aligned}
0 \leq \alpha_i^r \leq C \,, \\
y_i^r \left(w \cdot \phi(x_i^r) + b + w_r \cdot \phi_r(x_i^r) + b_r\right) - p_i^r + \xi_i^r \geq 0 \,, \\
\alpha_i^r \left(y_i^r \left(w \cdot \phi(x_i^r) + b + w_r \cdot \phi_r(x_i^r) + b_r\right) - p_i^r + \xi_i^r\right) = 0 \,, \\
0 \leq \beta_i^r \leq C \,, \\
\xi_i^r \geq 0 \,, \\
\beta_i^r \xi_i^r = 0 \,, \\
\alpha_i^r + \beta_i^r = C \,,
\end{aligned}$$

for $i = 1, \ldots, m_r,\ r = 1, \ldots, T$. The KKT conditions ensures that if we have a tuple $(w^*, w_r^*, b^*, b_r^*, \xi^*, \alpha^*, \beta^*)$ that fulfills the KKT conditions, then the tuple $(w^*, w_r^*, b^*, b_r^*, \xi^*)$ is the optimal solution of the primal problem and (α^*, β^*) is the optimal solution of the dual problem. In order to have an easier condition to check optimality, the following lemma, analogous to the Lemma 2.4.1, can be proved.

Lemma 3.3.1. *For each task r, define the following sets*

$$\begin{aligned}
I_{U_r}^+ &= \{i \in \mathcal{T}_r : y_i^r = 1 \wedge \alpha_i^r > 0\} \,, \\
I_{U_r}^- &= \{i \in \mathcal{T}_r : y_i^r = -1 \wedge \alpha_i^r < C\} \,, \\
I_{L_r}^+ &= \{i \in \mathcal{T}_r : y_i^r = 1 \wedge \alpha_i^r < C\} \,, \\
I_{L_r}^- &= \{i \in \mathcal{T}_r : y_i^r = -1 \wedge \alpha_i^r > 0\} \,;
\end{aligned}$$

and also define

$$\begin{aligned}
I_{U_r} &= I_{U_r}^+ \cup I_{U_r}^- \,, \\
I_{L_r} &= I_{L_r}^+ \cup I_{L_r}^- \,;
\end{aligned}$$

then, if the KKT conditions hold, $\forall u \in I_{U_r},\ \forall l \in I_{L_r}$,

$$w \cdot \phi(x_u^r) + w_r \phi_r(x_u^r) - y_u^r p_u^r \leq -b - b_r \leq w \cdot \phi(x_l^r) + w_r \phi_r(x_l^r) - y_l^r p_l^r \,. \tag{3.3.7}$$

Proof. The two cases $0 \leq \alpha_i^r < C$ and $0 < \alpha_i^r \leq C$ are proved separately. Note that although the two sets are not disjoint, every α_i^r is covered in one of them; then using the KKT conditions the two cases can be treated in the following way: the first case is $\alpha_i^r < C \implies \beta_i^r > 0 \implies \xi_i^r = 0$, which can be followed from the KKT conditions; then

$$y_i^r \left(w \cdot \phi(x_i^r) + w_r \phi_r(x_i^r) - y_i^r p_i^r + b + b_r\right) - p_i^r \geq 0 \,;$$

that is

$$\begin{aligned}
w \cdot \phi(x_i^r) + w_r \phi_r(x_i^r) - y_i^r p_i^r \geq -b - b_r \text{ if } y_i^r = 1 \ (I_{L_r}^+) \,, \\
w \cdot \phi(x_i^r) + w_r \phi_r(x_i^r) - y_i^r p_i^r \leq -b - b_r \text{ if } y_i^r = -1 \ (I_{U_r}^-) \,.
\end{aligned}$$

The second case is $\alpha_i^r > 0 \implies y_i^r \left(w \cdot \phi(x_i^r) + w_r \phi_r(x_i^r) - y_i^r p_i^r + b + b_r\right) - p_i^r + \xi_i^r = 0$, which can be followed by the KKT conditions; then

$$y_i^r \left(w \cdot \phi(x_i^r) + w_r \phi_r(x_i^r) - y_i^r p_i^r + b + b_r\right) - p_i^r \leq 0 \,;$$

that is

$$w \cdot \phi(x_i^r) + w_r \phi_r(x_i^r) - y_i^r p_i^r \leq -b - b_r \text{ if } y_i^r = 1 \ (I_{U_r}^+) \ ,$$
$$w \cdot \phi(x_i^r) + w_r \phi_r(x_i^r) - y_i^r p_i^r \geq -b - b_r \text{ if } y_i^r = -1 \ (I_{L_r}^-) \ .$$

□

This lemma provides an easy two-step rule for selecting (U_r^*, L_r^*); that is, the indices to update in the multi-task problem . The rule consists on preselecting first the pair of indices (U_r, L_r) of each task that violate most the KKT conditions; then the indices of the task with the most violating pair, noted $(U_{r^*}, L_{r^*}) = (U_r^*, L_r^*)$, are selected to be updated. More formally, notice first the following.

$$\Theta(\alpha) = \frac{\mu}{2} \langle w^*, \frac{1}{\mu} \sum_{r=1}^{T} \sum_{i=1}^{n_r} \alpha_i^r y_i^r \phi(x_i^r) \rangle + \frac{1}{2} \sum_{r=1}^{T} \langle w_r^*, \sum_{i=1}^{n_r} \alpha_i^r y_i^r \phi_r(x_i^r) \rangle - \sum_{r=1}^{T} \sum_{i=1}^{n_r} \alpha_i^r p_i^r \ ;$$

then taking derivatives with respect to α_i^r the result is

$$(\nabla \Theta(\alpha))_j^r = \frac{\partial \Theta(\alpha)}{\partial \alpha_i^r} = \langle w^*, y_i^r \phi(x_i^r) \rangle + \langle w_r^*, y_i^r \phi_r(x_i^r) \rangle - p_i^r \ .$$

Therefore, multiplying the partial derivative times y_i^r we get

$$y_i^r (\nabla \Theta(\alpha))_j^r = y_i^r \frac{\partial \Theta(\alpha)}{\partial \alpha_i^r} = \langle w^*, \phi(x_i^r) \rangle + \langle w_r^*, \phi_r(x_i^r) \rangle - y_i^r p_i^r \ ;$$

thus, the condition of the equation (3.3.7) can be written as

$$y_u^r (\nabla \Theta(\alpha))_u^r \leq y_l^r (\nabla \Theta(\alpha))_l^r, \ \forall u \in I_{U_r}, \ \forall l \in I_{L_r} \ .$$

Given this result, we define Δ_r as

$$\Delta_r(\alpha) = \max_{u \in I_{U_r}} (y_u (\nabla \Theta)_u) - \min_{l \in I_{L_r}} (y_l (\nabla \Theta)_l) \ , r = 1, \ldots, T \ ;$$

this definition implies that if $\Delta_r(\alpha) \leq 0 \ \forall r = 1, \ldots, T$, then by the Lemma 3.3.1 α is the optimal solution. However, if any $\Delta_r(\alpha) > 0$ then the KKT conditions are not met and therefore α is not optimal. Moreover any pair $u \in I_{U_r}, \ l \in I_{L_r}$ such that

$$y_u^r (\nabla \Theta(\alpha))_u^r - y_l^r (\nabla \Theta(\alpha))_l^r > 0$$

is called a 'violating pair'. With this information, the heuristic rule used is to select r^* in the following way:

$$r^* = \arg\max_r \Delta_r(\alpha) \ ,$$

and once r^* is selected, the indices are simply

$$U_r^* = U_{r^*} = \arg\max_{u \in I_{U_{r^*}}} (y_u (\nabla \Theta)_u) \ ,$$
$$L_r^* = L_{r^*} = \arg\min_{l \in I_{L_{r^*}}} (y_l (\nabla \Theta)_l) \ . \tag{3.3.8}$$

Given the definition of the selection rule, it shows that the indices selected are those that most violate the KKT conditions and thus the rule is called 'maximal violating pair' rule.

Until now, we have described the 'maximal violating pair rule', which uses information from the gradient in order to select U and L. However, a more efficient approach can be made. The main goal in order to achieve convergence faster is to minimize

$$\Psi(\delta) = \Delta\Theta(\alpha) = \Theta(\alpha + \delta) - \Theta(\alpha) \ ,$$

which, using (3.3.2), can be written as

$$\begin{aligned}\Psi(\delta) = \Delta\Theta(\alpha) &= \Theta(\alpha + \delta) - \Theta(\alpha) \\ &= \frac{1}{2}\delta(\tau)^t\hat{Q}\delta(\tau) - \left(\delta(\tau)^t\hat{Q}\alpha(\tau) + p\delta(\tau)\right) \ .\end{aligned}$$

Moreover, using the derivatives of $\Theta(\alpha)$ which are

$$\begin{aligned}\nabla\Theta(\alpha) &= \alpha^t\hat{Q} + p \ , \\ \nabla^2\Theta(\alpha) &= \hat{Q} \ ,\end{aligned}$$

we can write $\Psi(\delta)$, which is quadratic, in the following way

$$\Psi(\delta) = \Delta\Theta(\alpha) = \Theta(\alpha + \delta) - \Theta(\alpha) = \nabla\Theta(\alpha)\delta + \frac{1}{2}\delta^t\nabla^2\Theta(\alpha)\delta \ .$$

We have already seen in (3.3.5) that it can be expressed just in terms of δ_L:

$$\begin{aligned}\Psi(\delta) = \psi(\delta_{L_r}) = y_{L_r}\delta_{L_r}\left(y_{L_r}(\nabla\Theta)_{L_r} - y_{U_r}(\nabla\Theta)_{U_r}\right) + \\ \frac{1}{2}\delta_{L_r}^2\left(\frac{1}{\mu}\left\|\phi(x_{U_r}) - \phi(x_{L_r})\right\|^2 + \left\|\phi_r(x_{U_r}) - \phi_r(x_{L_r}\right\|^2)\right) \ .\end{aligned}$$

As in SMO, in (3.3.8) we are using just information about the first derivative. Working with the function $\Psi(\delta)$ and the result of (3.3.6) the following proposition can be used in this multi-task framework.

Proposition 3.3.1. *Let (U_r, L_r) be a violating pair such that*

$$\left(\frac{1}{\mu}\left\|\phi(x_{U_r}) - \phi(x_{L_r})\right\|^2 + \left\|\phi_r(x_{U_r}) - \phi_r(x_{L_r}\right\|^2)\right) > 0 \ ;$$

then $\Psi(\delta)$ has the optimal value

$$-\frac{1}{2}\frac{\left(y_{U_r}(\nabla\Theta)_{U_r} - y_{L_r}(\nabla\Theta)_{L_r}\right)^2}{\frac{1}{\mu}\left\|\phi(x_{U_r}) - \phi(x_{L_r})\right\|^2 + \left\|\phi_r(x_{U_r}) - \phi_r(x_{L_r}\right\|^2} \ . \tag{3.3.9}$$

Proof. The proof is analogous to the one given for Proposition 2.4.1. The difference is due to the fact that the quadratic term used for the multi-task approach is $\hat{Q}$ instead of Q; thus, the term corresponding to the norm is the following:

$$\frac{1}{\mu}\left\|\phi(x_{U_r}) - \phi(x_{L_r})\right\|^2 + \left\|\phi_r(x_{U_r}) - \phi_r(x_{L_r}\right\|^2 \qquad \square$$

Using Proposition 3.3.1 it is sufficient to choose (U_r, L_r) as a violating pair that minimizes (3.3.9), which, however, requires a quadratic cost search. The solution is to select a violating pair (U_r, L_r) for each task in the following way:

$$\begin{aligned} U_r &= \underset{u \in I_{U_r}}{\arg\max} \left(y_u (\nabla\Theta)_u\right) , \\ L_r &= \underset{l \in I_{L_r}}{\arg\min} \left(- \frac{\left(y_{U_r}(\nabla\Theta)_{U_r} - y_{L_r}(\nabla\Theta)_{L_r}\right)^2}{\frac{1}{\mu}\left\|\phi(x_{U_r}) - \phi(x_{L_r})\right\|^2 + \left\|\phi_r(x_{U_r}) - \phi_r(x_{L_r}\right\|^2} \right) . \end{aligned} \tag{3.3.10}$$

By selecting (U_r, L_r) in this way, the cost is linear since both steps consist of a linear search. Then, we simply select (U_{r^*}, L_{r^*}) as the pair from task r^* that maximizes the gain

$$\frac{\left(y_{U_r}(\nabla\Theta)_{U_r} - y_{L_r}(\nabla\Theta)_{L_r}\right)^2}{\frac{1}{\mu}\left\|\phi(x_{U_r}) - \phi(x_{L_r})\right\|^2 + \left\|\phi_r(x_{U_r}) - \phi_r(x_{L_r}\right\|^2} .$$

As in the standard SMO algorithm, GSMO makes use of the gradient information in order to choose the variables to update; thus, it is necessary to maintain the whole gradient at each step. The update is similar to the one seen in SMO. The main difference is that in GSMO we need to take α_U, α_L corresponding to the same tasks in order to fulfill the equality constraints. That is, the update is

$$\nabla\Theta(\alpha)^{\tau+1} = \nabla\Theta(\alpha)^{\tau} + Q\delta^{\tau} ,$$

where

$$\delta^t = (0, \ldots, 0, \delta_{U_r}, 0, \ldots, \delta_{L_r}, \ldots, 0) .$$

3.3.3 Computational Cost

Once we have seen the differences between standard SMO and GSMO for multi-task learning, we show in Algorithm 2 the entire *Dual Order GSMO* for a better understanding. Notice that we can divide Algorithm 2 in two parts, one for each loop. In the first loop the variables are initialized. For each task r, we compute the indexes I_U^r and I_L^r, the difference Δ^r, the pair of indexes U_r, L_r that maximizes the gain and the gain G_r itself. Once all this values have been computed for every task, the second loop is the one that actually performs the GSMO algorithm. In first place, we select the task r^* that maximizes the gain, then, we perform the update of the dual variables and the gradient. Finally, at the end of each iteration, the values of the indexes $I_U^{r^*}$ and $I_L^{r^*}$, the difference Δ^{r^*}, the pair of indexes U_{r^*}, L_{r^*} and the gain G_{r^*} are also updated for the next iteration.

With the goal of examining the computational cost of GSMO, we first study the cost of each iteration. As in the analysis made in SMO, we will go through each step. The Selection Step consists in two inner steps. In first place, the task with the maximal violating pair r^* is searched, which requires a search over all tasks. Once r^* has been selected, we perform two linear cost searches to find indices U_r and L_r, however we only take into account the patterns belonging to task r^*; that is, provided that we have $m = N/T$ patterns in average for each task the cost is $O(T+2N/T)$. The Update Step requires some comparisons task, which has an cost $O(1)$. The next step is the Gradient Update, which, as in standard SMO, consists in the multiplication of $\hat{Q}$ by the sparse vector δ, which has a cost of $O(2N)$. Finally, the computation of Δ_{r^*} requires checking in which sets $I_U^{r^*}$ or $I_L^{r^*}$ the new updated variables belong to, which has a cost $O(T)$. Therefore, the computational cost of iteration of GSMO is, in average, $O(T + 2N/T + 2N)$, which is greater than $O(3N)$ independently of T. Although we can compute the cost of each iteration, the convergence rate of this algorithm is left for further work.

Algorithm 2: GSMO

Data: matrix $\hat{Q}$, vector p, parameter C, vector T
Result: vector α^*
$\alpha \leftarrow \vec{0}$; $\nabla\Theta \leftarrow -p$;
for r, I_r *in* T **do**
  $I_U^r \leftarrow \{i \in I_r : y_i = 1 \wedge \alpha_i > 0\} \cup \{i : y_i = -1 \wedge \alpha_i < C\}$;
  $I_L^r \leftarrow \{i \in I_r : y_i = 1 \wedge \alpha_i < C\} \cup \{i : y_i = -1 \wedge \alpha_i > 0\}$;
  $\Delta^r = \max_{u \in I_U^r}(y_u(\nabla\Theta)_u) - \min_{l \in I_L^r}(y_l(\nabla\Theta)_l)$; // KKT conditions
  $U_r = \arg\max_{u \in I_U^r}(y_u(\nabla\Theta)_u)$;
  $L_r = \arg\min_{l \in I_L^r}\left(-\frac{\left(y_{U_r}(\nabla\Theta)_{U_r} - y_l(\nabla\Theta)_l\right)^2}{\hat{Q}_{U_rU_r} - 2\hat{Q}_{U_rl} + \hat{Q}_{ll}}\right)$;
  $G_r = \left(\frac{\left(y_{U_r}(\nabla\Theta)_{U_r} - y_{L_r}(\nabla\Theta)_{L_r}\right)^2}{\hat{Q}_{U_rU_r} - 2\hat{Q}_{U_rL_r} + \hat{Q}_{L_rL_r}}\right)$; // Gain
while $\{\Delta^r > tol\} \neq \emptyset$ **do**
  $r^* = \arg\max G_r$; // Selection Step
  $\lambda_{r^*} \leftarrow \frac{\left(y_{U_{r^*}}(\nabla\Theta)_{U_{r^*}} - y_{L_{r^*}}(\nabla\Theta)_L\right)}{\hat{Q}_{U_{r^*}U_{r^*}} - 2\hat{Q}_{U_{r^*}L_{r^*}} + \hat{Q}_{L_{r^*}L_{r^*}}}$;
  if $y_{L_{r^*}} == 1$ **then** $\lambda_{r^*} = \min(C - \alpha_{L_{r^*}}, \lambda_{r^*})$;
  else $\lambda_{r^*} = \min(\alpha_{L_{r^*}}, \lambda_{r^*})$;
  if $y_{U_{r^*}} == 1$ **then** $\lambda_{r^*} = \min(\alpha_{U_{r^*}}, \lambda_{r^*})$;
  else $\lambda_{r^*} = \min(C - \alpha_{U_{r^*}}, \lambda_{r^*})$;
  $\alpha_{L_{r^*}} \leftarrow \alpha_{L_{r^*}} + y_{L_{r^*}}\lambda_{r^*}$; $\alpha_{U_{r^*}} \leftarrow \alpha_{U_{r^*}} - y_{U_{r^*}}\lambda_{r^*}$; // Update Step
  $\delta = \vec{0}$;
  $\delta[L_{r^*}] \leftarrow y_{L_{r^*}}\lambda$; $\delta[U_{r^*}] \leftarrow -y_{U_{r^*}}\lambda$;
  $\nabla\Theta \leftarrow \nabla\Theta + \hat{Q}\delta$; // Gradient Update
  $I_U^{r^*} \leftarrow \{i \in I_{r^*} : y_i = 1 \wedge \alpha_i < C\} \cup \{i : y_i = -1 \wedge \alpha_i < C\}$;
  $I_L^{r^*} \leftarrow \{i \in I_{r^*} : y_i = 1 \wedge \alpha_i > 0\} \cup \{i : y_i = -1 \wedge \alpha_i > 0\}$;
  $\Delta^{r^*} = \max_{u \in I_U^r}(y_u(\nabla\Theta)_u) - \min_{l \in I_L^r}(y_l(\nabla\Theta)_l)$; // KKT conditions
  $U_{r^*} = \arg\max_{u \in I_U^{r^*}}(y_u(\nabla\Theta)_u)$;
  $L_{r^*} = \arg\min_{l \in I_L^{r^*}}\left(-\frac{\left(y_{U_{r^*}}(\nabla\Theta)_{U_{r^*}} - y_l(\nabla\Theta)_l\right)^2}{\hat{Q}_{U_{r^*}U_{r^*}} - 2\hat{Q}_{U_{r^*}l} + \hat{Q}_{ll}}\right)$;
  $G_{r^*} = \left(\frac{\left(y_{U_{r^*}}(\nabla\Theta)_{U_{r^*}} - y_{L_{r^*}}(\nabla\Theta)_{L_{r^*}}\right)^2}{\hat{Q}_{U_{r^*}U_{r^*}} - 2\hat{Q}_{U_{r^*}L_{r^*}} + \hat{Q}_{L_{r^*}L_{r^*}}}\right)$; // Gain Update

3.4 Single Bias Multi-task SVM

In Section 3.2, more concretely in (3.2.1), we have introduced the multi-task learning SVM, where multiple bias $b + b_r$ are used. When we go to the dual problem (3.2.8), multiple equality constraints are obtained from the multiple bias. That is the reason why GSMO is developed: it offers a variation of SMO that can deal with an equality constraint for each task. In order to be able to use the standard SMO algorithm, a small modification can be made to the problem defined in (3.2.8). Instead of using multiple biases $b + b_r$, a single

bias b is used for all tasks; that means the primal problem is then the following:

$$\begin{aligned} \underset{w,w_r,\xi}{\arg\min} \quad & J(w,w_r,\xi) = C\sum_{r=1}^{T}\sum_{i=1}^{m}\xi_i^r + \frac{1}{2}\sum_{r=1}^{T}\|w_r\|^2 + \frac{\mu}{2}\|w\|^2 \\ \text{s.t.} \quad & y_i^r(w\cdot\phi(x_i^r) + w_r\cdot\phi_r(x_i^r) + b) \geq p_i^r - \xi_i^r,\ i = 1,\ldots,m_r;\ r = 1,\ldots,T, \\ & \xi_i^r \geq 0,\ i = 1,\ldots,m_r;\ r = 1,\ldots,T\ . \end{aligned} \tag{3.4.1}$$

That way, developing the Lagrangian and taking derivatives as done before, the multi-task dual problem is the following:

$$\begin{aligned} \underset{\alpha}{\arg\min} \quad & \Theta(\alpha) = \frac{1}{2}\alpha^t\hat{Q}\alpha - p\alpha \\ \text{s.t.} \quad & 0 \leq \alpha_i^r \leq C,\ i = 1,\ldots,m_r; r = 1,\ldots,T \text{ (box constraints)}\ , \\ & \sum_{r=1}^{T}\sum_{i=1}^{n_r}\alpha_i^r y_i^r = 0,\ \text{(unique equality constraint)}\ . \end{aligned} \tag{3.4.2}$$

It is easy to see that in this problem SMO can be applied directly, without any modification. This single-bias multi-task problem will be useful in order to carry out the experiments in Chapter 4.

Chapter 4

Experiments

This chapter presents the experiments carried out and it provides an explanation of the tools used. Section 4.1 is an overview of LIBSVM, one of the most important libraries for using SVMs, and especially `Scikit-learn` that wraps the functionality of SVMs and it is the library used in this work. In this section we will also discuss the *mtlSVM* class we have created and some of its implementation details. Section 4.2 shows multiple experiments using a popular multi-task dataset about school students. The purpose of these experiments is to compare our results with previous ones as well as to get a better understanding of the implications of the multi-task approach in a simple problem. Finally, in Section 4.3 we perform an experiment using real solar energy data from the islands of Majorca and Tenerife in Spain. We compare the results obtained with a single-task SVR and multi-task SVR when using the data from both islands.

4.1 Implementation Details

In this section we will describe the implementation of the multi-task SVM that has been used. In order to do that we have to know the main implementations of the methods to solve SVM problems; that is LIBSVM and `Scikit-learn`, which is implemented in Python but based on LIBSVM. After the presentation of these libraries, we will explain the implementation of the multi-task SVM, which was first presented in [2].

4.1.1 LIBSVM and `Scikit-learn`

LIBSVM [17] is the most popular library that implements the SMO algorithm for kernelized support vector machines, and it supports both classification and regression. It is written in C++ and it is free and open software. Its code has been widely reused and bindings for other languages have been developed. In this work there is a particular interest in the implementation developed as a part of `Scikit-learn` [18]. `Scikit-learn` is a free and open library for machine learning written in Python. It features several machine learning methods for both classification and regression, we are interested on its SVM implementation though. In `Scikit-learn`, a distinction is made between the SVM for classification, called Support Vector Classifier (SVC) and the SVM for regression called Support Vector Regressor (SVR). The SVC and SVR are different classes and since one is a classifier and the other a regressor, there are some differences between them. We will first take a look at the SVC and SVR implementations. Every estimator of `Scikit-learn` inherits from `BaseEstimator`, so SVC and SVR as every other estimator class of `Scikit-learn`, must implement the following methods:

- `_init_()`: The initialization method of Python classes. It receives all the meta-information about the model, including its hyper-parameters. This is where the hyper-parameters of the models are set. The parameters in which we are interested for the goal of this work are the following:
  - `C` : Penalty parameter of the error term.
  - `kernel` : Specifies the kernel type to be used in the algorithm. It can be `linear`, `polynomial`, `rbf` or `precomputed`. For this work, in order to compute the multi-kernel Gramm Matrix $\hat{Q}$, the `precomputed` option has been used.
  - `gamma` : Kernel width coefficient for RBF kernel.
  - `epsilon`: Used only in SVR, it is the parameter that tunes the width of the error-tolerant tube within which there is no penalty for points in the training sample.

- `fit(X, y)`: This method receives the patterns as the rows of the matrix `X` and its corresponding targets in the vector `y`. It performs the necessary computations in order to have the model adjusted to the dataset. In the case of SVM, it features the SMO algorithm, as described in LIBSVM, to find the dual solution α^*. It also keeps the support vectors, which are necessary to make predictions.

- `predict(X)`: This method returns the prediction for the label or target made by the model for each new unseen example given. In the case of SVM, for a given unseen example $\hat{x}$ it returns

$$w^* \cdot \phi(\hat{x}) + b = \sum_{i \in \mathcal{I}} y_i \alpha_i^* \phi(x_i) \cdot \phi(\hat{x}) + b = \sum_{i \in \mathcal{I}} y_i \alpha_i^* k(x_i, \hat{x}) + b \, ,$$

  where $\mathcal{I}$ are the indices of the support vectors of the training sample.

- `score(X, y)`: This method, like the previous one, computes the prediction of the model for each new example, and then, it computes a score function measuring the adequacy of the prediction to the real values of the target or label. In the case of classification the accuracy score is used, while for regression is the R2 score.

Aiming to achieve a high efficiency and accurate results, the `Scikit-learn` implementations of SVM have been used. However, in order to adapt the multi-task approach, a wrapper class that makes use of `Scikit-learn` implementation has been developed.

The software presented in this subsection implements the SMO algorithm but not the GSMO algorithm explained in Section 3.3. The solution proposed is to use the single-bias multi-task SVM presented in Subsection 3.4. By defining the multi-task SVM this way, we obtain a model that is still valid for a multi-task framework though may not be as flexible as the one defined in (3.2.1). Nevertheless, since the bias is the average of the target, we can center the targets of each task around zero; therefore, the bias term is not so important and the weight vector $w + w_r$ that is different for each model should be able to bring out the differences between tasks. Moreover, the main advantage of this model with a single bias is that we can use the software included in `Scikit-learn`. The only modification that needs to be made is the way the `kernel` option is selected in order to use $\hat{Q}$. Since the `Scikit-learn` software has the option of using a precomputed kernel matrix, this is easy to do: we just precompute the entire matrix $\hat{Q}$ and then we call the `Scikit-Learn` methods.

4.1.2 The `mtlSVM` Class

The `mtlSVM` class is mainly a wrapper class that has a `Scikit-learn` SVM object as an attribute. The idea is to make use of the 'precomputed' kernel option in order to compute the multi-task Gramm matrix $\hat{Q}$ and then calling the `fit` method of the `Scikit-learn` object with this matrix. To be compatible with the `Scikit-learn` framework, the `mtlSVM` class implements the methods `fit`, `predict` and `score`. Moreover, imitating `Scikit-learn`, the class used for the experiments is `mtlSVR`, which inherit from `mtlSVM`. The `mtlSVR` class has an object of the `Scikit-learn SVR` class. Then, the parameters concerning the model that have to be provided for the `mtlSVM` class are:

- `C`: Penalty parameter of the error term.
- `ckernel` : Specifies the kernel type to be used for the common part of the model in the algorithm.
- `skernel`: Specifies the kernel types to be used for the specific part of the model in the algorithm. If a single value is provided, every task will use the same kernel for the specific part. If a list is provided, each task is assigned its corresponding kernel. When using multiple kernels in the specific part, it is assumed that the kernels are ordered in such a way that the first kernel corresponds to the first task (when the tasks labels are alphabetically sorted).
- `cgamma` : Kernel width coefficient for the common part kernel when the RBF kernel is used.
- `sgamma` : Kernel width coefficient for the specific part kernel when the RBF kernel is used. If there are multiple kernels for the specific part, an array with one value of gamma for each kernel is expected.
- `epsilon`: It tunes the width of the error-insensitive tube within which there is no penalty for points in the training sample.
- `mu`: Common part regularization parameter. It tunes the shrinking made to the matrix Q which has the information of the common part of the model. With large values of `mu`, the models obtained for each task are more different among them.

The way to use the multi-task kernel matrix $\hat{Q}$ is to compute the whole matrix and then to use the `precomputed` option for the kernel and call the `fit` or `predict` method of the `Scikit-learn` SVM object. With this wrapper class implementation, two experiments have been carried out. The first one, described in Section 4.2, tries to replicate the experiment shown in [1], while the second one, explained in Section 4.3, aims to predict the solar energy production of the islands of Mallorca and Tenerife.

4.2 A first application: Prediction of school grades

The goal of this section is two-fold: in the first place to check the validity of the `mtlSVM` implementation; and secondly, to get a better visualization of the consequences of using a multi-task learning approach with SVMs. In order to validate our implementation, the experiment tries to replicate the one carried out in [1].

4.2.1 Dataset Description

The dataset [19] comes from the Inner London Education Authority (ILEA) and consists of 15,362 examination records from 139 secondary schools during the years 1985, 1986 and 1987. These data, which have been used to study the effectiveness of schools [20], is a popular one among the works in multitask learning [3, 1, 4]. The original 9 features and the task variable are the following:

Year The year in which the examination record was obtained. It takes 3 values (1985=1, 1986=2, 1987=3), so it has been encoded into three new binary variables.

School This is the identifier of the school or, in our approach, the task identifier. It takes values from 1 to 139.

Exam Score This is the numeric score obtained in the exam; that is, the target variable.

%FSM This is a percentage, it represents the percentage of students eligible for free school meals.

%VR1 band The percentage of students in school in VR band 1. That is, the students are divided into three bands as a result of its score in a Verbal Reasoning test, being band 1 the one with the best results and band 3 the band with the worst ones. This feature shows the percentage of students of the school that are in the first VR band.

Gender This takes just two values (Male=0, Female=1).

VR band of student The Verbal Reasoning band in which the student is included. It takes three values, one for each band, so it has been encoded into three binary variables.

Ethnic group of student It takes 11 values for different ethnic groups (ESWI = 1; African = 2; Arab = 3; Bangladeshi = 4; Caribbean = 5; Greek = 6; Indian = 7; Pakistani = 8; S.E.Asian = 9; Turkish = 10; Other = 11). ESWI stands for students born in England, Scotland, Wales or Ireland. It has been encoded into eleven binary variables.

School Gender It takes three values (Mixed=1; Male=2; Female=3) so it has been encoded into three binary variables.

School Denomination It takes three values (Maintained=1; Church of England=2; Roman Catholic=3) so it has been encoded into three binary variables.

The final result is a dataset with 27 features where one is the task, 2 of them are numeric, and the remaining 24 are binary. For the experiments we have also normalized the features to have zero mean and a standard deviation of 1. In the problem defined in [1] there is no bias. Because of that we have centered the target variable around zero as well in order to make the bias less relevant. Although this is a multi-task dataset, the number of examples is not the same for every task as it can be seen in Figure 4.2.1. Due to the fact that some tasks have a number of examples that is too low, we discard the possibility of using one SVM for each task. Instead of that, we will compare a global single-task SVM and a multi-task SVM.

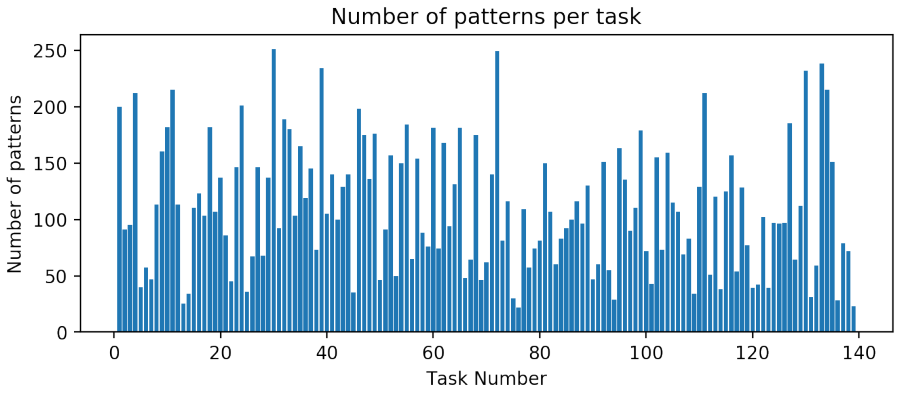

Figure 4.2.1: Each bar height represents the number of examples in the task that corresponds to its x coordinate.

4.2.2 Results

The main goal of this experiment is to compare our results with those achieved in [1] and to illustrate the differences between the single-task and multi-task models. For this comparison, the single-task and multi-task model's hyperparameters have been selected using cross validation. First we split the data in train and test using the `train_test_split` function from `Scikit-learn` with a percentage of 20% for test. Then, for the cross validation the `StratifiedKFold` class is used, using the task for the stratification of 10 folds. Finally, a linear kernel is used and the score chosen for the validation is the R2 score, since we are trying to compare our results with those obtained in [1]. For the single-task SVR, the following logarithmic grid has been explored:

- $C \in \{10^k : -2 \leq k \leq 2\}$.
- $\epsilon \in \{2^{-k}\sigma : 1 \leq k \leq 4\}$, where σ denotes the standard deviation of the target variable.

The best parameters for the SVR are $C = 10$ and $\epsilon = 6.3702 = \sigma/2$. Note that both fall in the middle of the grid used. The R2 score achieved in test by the SVR using these parameters is 0.3394. For the multi-task SVR, we add a new dimension to the grid for μ, that is, the grid explored is:

- $C \in \{10^k : -2 \leq k \leq 2\}$.
- $\epsilon \in \{2^{-k}\sigma : 2 \leq k \leq 5\}$ where σ denotes the standard deviation of the target variable.
- $\mu \in \{0.1, 0.5, 1, 2, 10, 1000\}$ which are the values used in [1] with the addition of 0.1 in order to check that the maximum score is achieved with $\mu = 0.5$.

The best parameters for the multi-task SVR are $C = 0.1$, $\epsilon = 6.3702 = \sigma/2$ and $\mu = 0.5$. The R2 score achieved with these parameters is 0.3775. Moreover, the optimal μ is the same as the one obtained in [1].

In Figure 4.2.2 the real value and its prediction of each point in test are shown. It is noticeable that the predictions are not perfect and the points do not lie close to the identity line, which is sensible since the R2 scores obtained are not very close to 1. The multi-task model appears to obtain a more balanced result than the single-task model though.

However, since we are in a multi-task framework, a global comparison does not show the difference between the models. In order to have a better understanding we compare

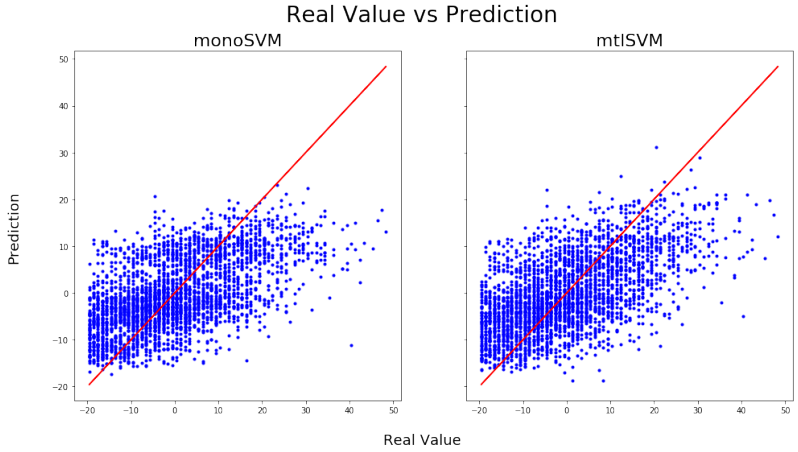

Figure 4.2.2: Both graphics show the real value in the x-axis and its prediction on the y-axis. The left one uses the prediction from the single-task SVR and the right one those from the multi-task SVR. The identity line, what would be a perfect prediction, is shown in red.

the results task by task using the R2 score. We can see the values in Figure 4.2.3, where there are 87 tasks where the multi-task model obtain a better result for 52 in which the single-task model performs better. It is interesting to notice that the advantage of the multi-task approach seems more evident for tasks with less examples in the training phase. This behaviour can be explained taking into account the characteristics of both models: with the single-task SVM we find a single weight vector w that tries to be good for every task; that is, we look for a w that minimizes a certain loss over all the points from any task. Assuming that examples from the same task are similar, the model will focus on those tasks with a larger number of points. That way, the total error will be lower but the model will not be adapted to those tasks with a low number of patterns. The multi-task model, on the contrary, trains one weight vector w_r for every task, and each one has an effect only on the patterns of such task. That way, every task will have its vector w_r adapted to its own patterns, not just those with a larger number of examples. And moreover, it is not that the vectors w_r of tasks with few examples will be trained using less examples and therefore, being less reliable. Every vector w_r uses every example from all the tasks in the training phase, but it puts its focus on the examples from its own task.

Although the cross validation has been carried out using the R2 score for comparisons, we are also interested in the Mean Absolute Error (MAE) of the models. Using the same parameters as before, the single-task SVM obtains a MAE of 8.226 while the multi-task SVM obtains a global MAE of 8.039. Again, since we are also interested in the results isolated by task, we perform the same experiment, where the MAE difference is represented for each task. The graphic shown in Figure 4.2.4 sheds a similar result as the one obtained for the R2 score, there are 90 tasks where the multi-task model achieves a better result and 49 where the single-task model is better. Moreover, the advantage of the multi-task approach looks more evident in tasks with fewer examples. Note that in this case, the MAE is better when it is low, therefore, if the difference is negative it means that the error made by the multi-task SVR is smaller.

Finally, the fact that we have chosen a linear kernel is interesting because using (3.2.4)

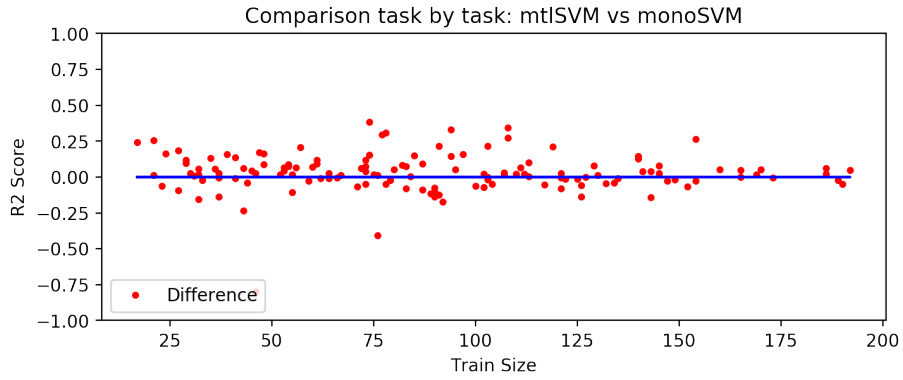

Figure 4.2.3: The points in red represent the difference of R2 score between multi-task SVR and single-task SVR in terms of the number of patterns of each task used in training. The blue line is set at zero, red points above the blue line mean that the multi-task SVR obtains a better score in those tasks. The inverse happens for points below the blue line.

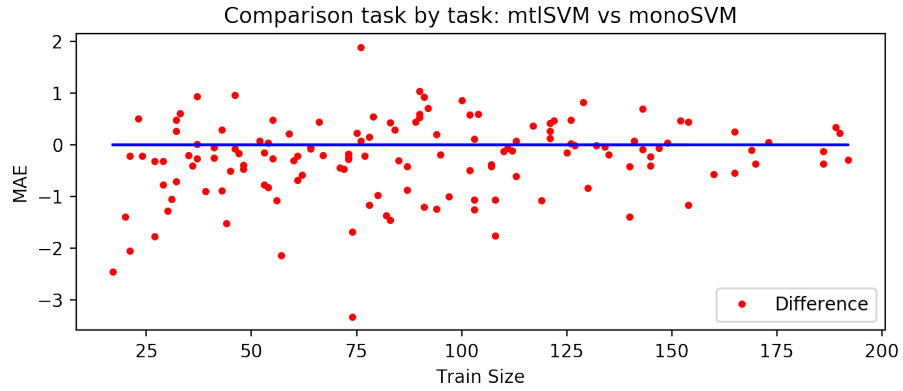

Figure 4.2.4: The points in red represent the difference of MAE between multi-task SVR and single-task SVR in terms of the number of patterns of each task used in training. The blue line is set at zero, red points above the blue line mean that the multi-task SVR obtains a larger MAE in those tasks. The inverse happens for points below the blue line.

we can compute directly the weight vectors w_r as

$$w_r = \sum_{i=1}^{n_r} \alpha_i^r y_i^r x_i^r$$

for every task, as well as the common vector w using (3.2.3), which results in the following:

$$w = \frac{1}{\mu} \sum_{r=1}^{T} \sum_{i=1}^{n_r} \alpha_i^r y_i^r x_i^r \ .$$

Moreover, in the same way we can also compute the single vector w of the single-task model using (2.3.11). Since the dimension of this problem is not too large and the scale of every feature is similar, it is possible to plot the weights vector w and w_r of the multi-task problem as well as the unique vector w of the single task problem. The result is represented in Figure 4.2.5 and it is easy to observe that the common part w lies in the middle of the blue area, that represents the interval between the minumum and maximum value of the

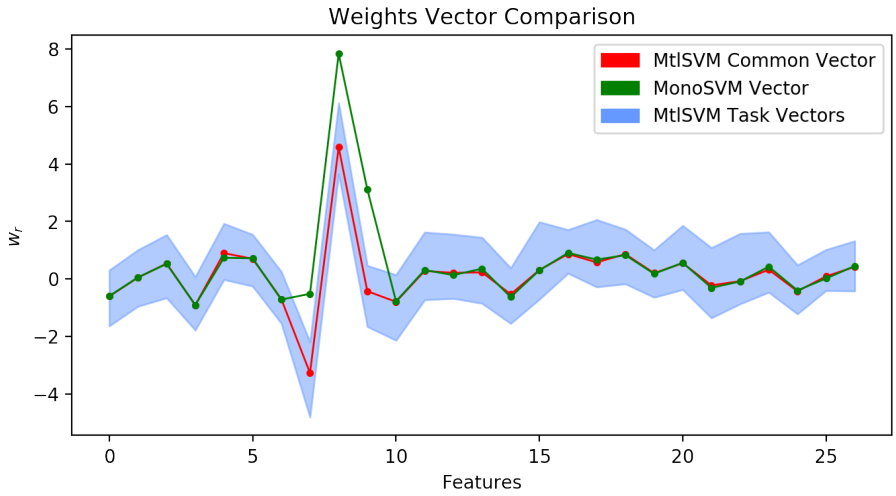

Figure 4.2.5: The x-axis represents the 28 features used in this problem. The red line is the common part w of task vectors in the multi-task approach. The blue area indicates the minimum and maximum in which every task vector v_r is in. Finally, the green line represents the vector w of the single-task SVR.

task vectors v_r; thus, it seems that in fact each task vector v_r is the common vector w plus a small deviation w_r that relies on the examples from such task. We can also notice that the green line representing the weight vector of the single-task SVR looks similar to the common part of the multi-task approach except at features 7, 8 and 9, which corresponds to the VR1 band features. This makes sense since both vectors are constructed using all the examples from the dataset; and since the VR1 band is a "task feature" that might be related to the test results, the single-task model gives them a greater importance than the multi-task models. The multi-task models can decide whether in its school those variables are that important to the test results. This election is made by making the common part smaller for these variables and adapting its specific part. It's clear in the figure that they are more flexible since it can adapt each vector v_r to its corresponding task by adding to the global vector a small specific deviation.

4.3 Energy Forecasting

Renewable energy relevance is rapidly increasing, and solar energy might be the most popular one, especially photovoltaic energy. Photovoltaics converts the radiation of the sunlight into electricity, thus, providing a clean and renewable source of energy. However, it is also a relatively novel source, which makes it important to develop methods to forecast the solar energy production which would ease its management. In fact, multiple works in machine learning have adopted this goal and have predicted solar energy in different parts of the world. In order to test the efficiency of the multi-task SVM approach with solar energy, we have artificially created a multi-task problem. Majorca and Tenerife are two islands of Spain, but they are not close. Majorca, which is part of the Balearic Islands is in the Mediterranean sea, while Tenerife, part of the Canary Islands, is in the Atlantic Ocean, more than $2,500$ km away. Both islands produce photovoltaic energy and previous works have been carried out aiming to predict the solar production in each one. However, in order to adapt the problem to our multi-task framework, we have combined the data from both islands and have treated them as two different tasks of the same multi-task problem.

Once the data is ready for the multi-task problem, we want to compare the multi-task SVM defined in (3.4.1) and a single-task SVM using data from both islands at the same time.

The data used has been taken from the Numerical Weather Prediction (NWP) provided by the ECMWF. The variables used are taken directly or have been constructed from the originals; they are the following:

1. Total cloud cover, a number from 0 to 1 (`TCC` in the nomenclature of the `ECMWF`).
2. Temperature at 2 meters (`T2M` in the nomenclature of the `ECMWF`).
3. Surface net solar radiation (`SSR` in the nomenclature of the `ECMWF`).
4. Surface solar radiation downwards (`SSRD` in the nomenclature of the `ECMWF`).
5. Module of the wind speed at 10 meters (`v10` in the nomenclature of the `ECMWF`).

These variables are provided for multiple coordinates. The Earth surface is divided in a grid of resolution 0.125^{o} and for each grid point the five variables can be taken. In the temporal dimension the data has an hourly resolution; that is, from each hour we have a value for each variable.

In order to adapt the data to the multi-task problem defined in (3.4.2) we have made some adjustments. In first place, since we have five features for each grid point, the total number of features depend on the size of the grid used. For this reason the rectangular grid used for the islands has the same dimensions for both. That is, we use a rectangle of 2 degrees of longitude and 1 degree of latitude more or less centered at each island. The one used for Majorca has its north-east corner in the coordinates $(2^{\text{o}}, 40^{\text{o}})$, and the south-west corner has coordinates $(4^{\text{o}}, 39^{\text{o}})$. The grid used for Tenerife has coordinates for the north-west and south-east corners of $(-17.5^{\text{o}}, 28.75^{\text{o}})$ and $(-15.5^{\text{o}}, 27.75^{\text{o}})$ respectively. The selection of these grids result in the same number of features for both tasks, which was an assumption made for (3.4.2). In particular, since our grid has $153 = 17 \times 9$ grid points, and five variables are provided for each point, the number of features used is $765 = 153 \times 5$. Moreover, the solar energy power installed in Tenerife is 107.68 MW while in Majorca it is 72.46 MW. In order to make the tasks more homogeneous and use a single-task SVM for both islands, the production, which is the target variable, has been normalized and it has been expressed as a percentage of the installed power. Since we are interested in SVR based models we have also scaled the data to the $[0, 1]$ range.

The temporal period used in this work are the years 2013, 2014 and 2015; therefore, for each island we have $26,280 = 3 \times 365 \times 24$ patterns. However, it is also important to note that the forecast of solar production is a problem that is interesting during those hours when there is sunlight. In Figure 4.3.1 we can observe that Majorca has sunlight from 06:00 to 20:00 UTC, while Tenerife has it from 08:00 to 21:00 UTC. The time period selected for our model is then from 06:00 to 21:00, since outside this region the solar production will be zero. Then, the number of patterns used for each island is $17,520 = 3 \times 365 \times 16$.

The models considered for this experiment are:

- *majSVR*, which is a single-task SVR just for data of Majorca.
- *tenSVR*, which is a single-task SVR just for data of Tenerife.
- *monoSVR*, which is a single-task SVR that uses data from both Majorca and Tenerife.
- *mtlSVR*, which is a multi-task SVR that uses data from both Majorca and Tenerife.

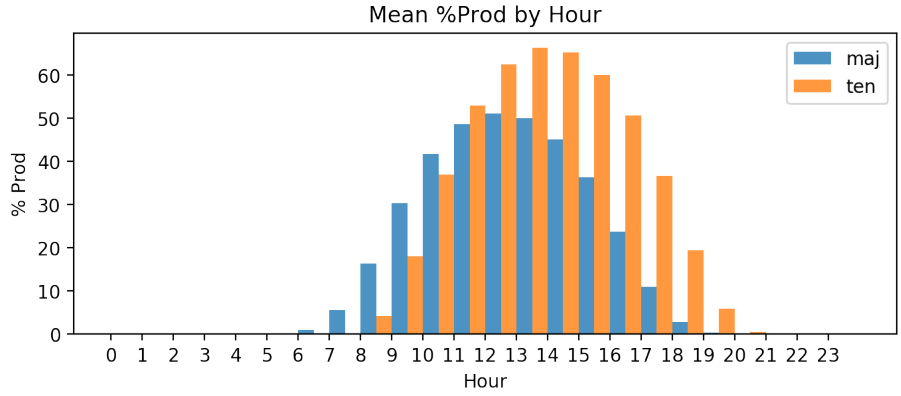

Figure 4.3.1: The average production as a percentage of the installed power each hour is represented in blue for Majorca and in orange for Tenerife. The hours shown in the x-axis corresponds to UTC.

The parameters for each model have been selected using a grid search in which we use 2013 for train and 2014 for validation using MAE as the validation criterion; then, we have $5,840 = 365 \times 16$ patterns for *majSVR* and *tenSVR* for both train and test, while we have $11,680 = 2 \times 365 \times 16$ for models *monoSVR* and *mtlSVR*. Once the best model in validation has been chosen, its accuracy its tested with the data from the year 2015; therefore, we use $5,840$ test patterns for *majSVR* and *tenSVR* and $11,680$ for *monoSVR* and *mtlSVR*.

The grid used for *majSVR*, *tenSVR* and *monoSVR* is the following:

- $C \in \{10^k : -5 \leq k \leq 6\}$.
- $\epsilon \in \{2^{-k}\sigma : 1 \leq k \leq 6\}$, where σ denotes the standard deviation of the target variable.
- $\gamma \in \{\frac{1}{d}4^k : -2 \leq k \leq 3\}$, where d are the number of features used, in our case $d = 765$.

For the *mtlSVR* model the same grid for C, γ and ϵ has been used, while the range for the multi-task parameter μ is the following:

- $\mu \in \{4^k - 1 \leq \mu \leq 3\}$.

The optimal parameters chosen for each model are shown in Table 4.3.1. In order to compare the models, we train each one setting its parameters to the optimal values found and using the years 2013 and 2014 for training. Then, we compute the Mean Absolute Error (MAE) of each model in the year 2015, which we use for testing purposes. The models *majSVR* and *tenSVR* are used to make predictions for Majorca and Tenerife respectively, while *monoSVR* and *mtlSVR* are used for both islands.

From the results shown in Table 4.3.2 we can draw two conclusions: in the first place, the multi-task model does not perform better than a single-task model for each task, that is, a model for each island. Neither in Majorca or Tenerife the *mtlSVR* model obtains a better result than *majSVR* or *tenSVR* respectively. This can be explained when we analyze the problem we are studying; on one hand, the tasks are not very related because each island is far apart from the other; on the other hand, we have all the data we need since we are using two full years of training to predict the solar production of one year. In other words, both the *majSVR* and *tenSVR* have all the useful information at their disposal in

Model	C	ϵ	γ	μ
majSVR	256	0.00574	0.00523	
tenSVR	64	0.02797	0.00523	
monoSVR	4096	0.00646	0.0208	
mtlSVR	4096	0.00646	0.00523	0.25

Table 4.3.1: Optimal parameters for the models considered.

	majSVR	*tenSVR*	*monoSVR*	*mtlSVR*
Majorca	6.237%	-	6.921%	6.483%
Tenerife	-	4.684%	5.283%	5.413%
Average	6.237%	4.684%	6.102%	5.948%

Table 4.3.2: MAE of each model in Majorca, Tenerife and in average.

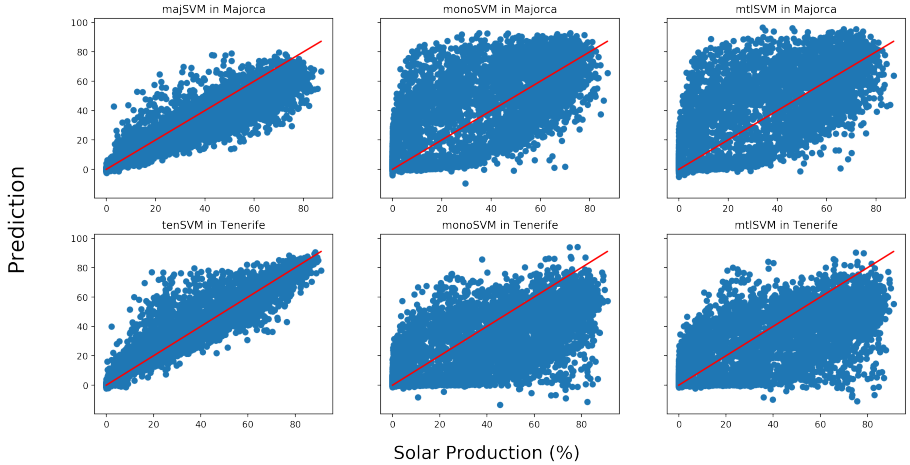

Figure 4.3.2: The blue dots are the production against prediction while the red line is the identity. The real solar production is represented as the x-coordinate and the prediction for the different models are represented in the y-coordinate. The first row is for Majorca and the second one for Tenerife.

order to learn the solar production, while the *mtlSVR* can use additional information from the other task that is not so useful because the tasks are not fully related to each other. If we had incomplete data for one of the islands, for example if we could not collect data from the winter in Tenerife; then, the additional information from the winter in Majorca would be more useful than the lack of information. However, when we use complete data, the data from Tenerife are more useful for Tenerife than data collected in Majorca. In the graphics shown in Figure 4.3.2, where the real production is compared with each model prediction, we can observe that individual models obtain clearly a better result. Another aspect that can explain this result is the fact that the features used represent a metereological variable in a certain geographic point. That means that the same feature in both islands may represent different things. For example, a group of features that correspond to a point in the island itself for Majorca could represent the same group of variables but placed in the sea for Tenerife. A consequence of this is that it is more difficult for the multi-task model since it has to deal with features that may represent different aspects.

The second conclusion is that the multi-task model performs better than a global single-task model for both islands, as it can be seen when comparing *mtlSVR* and *monoSVR*. This fact is supported by the greater flexibility of the multi-task model. Both models are given information that may be redundant or even unnecessary; however, the multi-task SVR generates two models, once for each task, which makes it possible for each task to rely stronger on its own patterns. On the other way, the single-task SVR does not make any difference between patterns from one island or the another; thus, giving too much importance to data from the other island may not be helpful.

Chapter 5

Discussion and Further Work

In this work we had two primary goals. In the first place we wanted to give an overview of the multi-task learning SVRs following the works of [1, 2]. Moreover, we wanted to show the connection between both works and use a formulation that makes this connection visible. This has been done in Chapter 3, where the multi-task SVMs are described as a variant that solves multiple tasks at once, training one common part of the models to all tasks, and one specific part of the models for each task. In Section 3.1 we have presented the approach called *Regularized Multi-task Learning*, introduced in [1], where we use a linear kernel for every task model and we omit the bias. This way the model is less flexible; and, since it is linear, it lacks the power of the traditional SVMs, in which we solve the maximum separability problem in an infinite space. In Section 3.2 we present the *MTLSVM* that solves these issues. In this formulation introduced in [2] the multi-task SVMs can use a non-linear kernel, and moreover, each task can use a different kernel. Also, in this approach, a bias is introduced for each task. With these additions, these models are more flexible and the multi-task SVMs are at least as expressive as the traditional single-task SVMS. In this work, the formulation of both approaches has been changed in order to make its connection visible, and we have shown that under certain circumstances the *MTLSVM* approach can be seen as a generalization of *Regularized Multi-task Learning*. We also present the algorithm GSMO developed in [2] that is used to train the *MTLSVM*. However, since the implementation of this algorithm is not trivial, in Section 3.4 we have introduced a hybrid model that uses a single bias for all tasks but it can use different kernels. However, the expressive power of this model may not be the same as the original *MTLSVM* and we leave for further work the implementation of GSMO using LibSVM.

Our second goal was to compare the results obtained by single-task SVRs and the introduced multi-task SVRs. Note that the multi-task SVRs used are those presented in Section 3.4. This comparison is carried out in Chapter 4, where we use two datasets: the first one that contains data of high school students and their performance, and the second one that combines the solar production data of two islands in order to have a multi-task problem. From the school experiment we can draw some conclusions. We have seen that the results obtained by the multi-task SVR are better than those obtained with a single-task SVR using the data from all tasks, both when comparing the R2 score or the MAE. Moreover, it is not just the overall result that is better, but when we compare the results task by task, we found that in the majority of the tasks the multi-task model has obtained a better result. Additionally, in order to compare the models we have depicted the weight vectors of each multi-task and single-task models. The graphic of Figure 4.2.5 shows that both the single-task weight vector and the common part of the multi-task weight vector are very similar despite the VR1 band feature. This feature measures the percentage of the

student of the school that fall in the first band of the test called 'verbal reasoning', which may be correlated to the results of the students in other tests. Nevertheless, the flexibility of the multi-task model allows small deviations from the common part, even giving less importance to the VR1 band feature when necessary; thus, providing a better result at the corresponding task. When we look at the experiments with the solar production data, we see similar results. In the first place, we also see that the multi-task model outperforms the single-task model using the data from all the tasks. However, we also observe that training one individual SVR for each task obtains better results than the global multi-task approach. This behaviour can be explained from the fact that the tasks are not totally related, since there is a great geographic separation between the islands. Moreover, the original features used correspond to one geographic point of a grid centered around each island; that means that the same feature can have a different meaning in each task. Although the kernel used is Gaussian, this fact may affect the multi-task results. We have also noted that for both tasks we have hourly data, that is, there is enough data to properly train an SVR. One of the main advantages of the multi-task learning is that when we do not have enough data to train a model for each task, we can combine all the data and train a multi-task model. Since the solar production problem is not the case, we are not providing a proper scenario that allows the multi-task SVR to show its full potential. Hence, we propose for further work to test the multi-task SVMs with a dataset that is more adequate for its characteristics.

To sum it up, in this work we provide the theoretical framework of the multi-task SVMs and we give an insight of its characteristics. The multi-task SVMs have proved to be competitive models that can perform well in multi-task problems, both when the tasks are similar or different from each other. However, in this work we have not developed the complete *MTLSVM* and moreover, we do not use a dataset that highlights the benefits of the multi-task SVMs. For that reason, we propose as future work to solve these issues in order to gain a better view of the power of multi-task SVMs.

Bibliography

[1] Theodoros Evgeniou and Massimiliano Pontil. Regularized multi–task learning. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 109–117. ACM, 2004.

[2] Feng Cai and Vladimir Cherkassky. Generalized SMO algorithm for SVM-based multitask learning. *IEEE transactions on neural networks and learning systems*, 23(6):997–1003, 2012.

[3] Tadej Janež, Jure Žabkar, Martin Možina, and Ivan Bratko. Learning Faster by Discovering and Exploiting Object Similarities. *International Journal of Advanced Robotic Systems*, 10(3):176, 2013.

[4] Bart Bakker and Tom Heskes. Task clustering and gating for bayesian multitask learning. *Journal of Machine Learning Research*, 4(May):83–99, 2003.

[5] Frank Rosenblatt. *The perceptron, a perceiving and recognizing automaton Project Para*. Cornell Aeronautical Laboratory, 1957.

[6] BA J Mercer. Xvi. Functions of positive and negative type, and their connection the theory of integral equations. *Phil. Trans. R. Soc. Lond. A*, 209(441-458):415–446, 1909.

[7] Michael Reed and Barry Simon. Methods of modern mathematical physics, vol. iii: Scattering theory. *New York, San Francisoco, London*, 1979.

[8] Nello Cristianini and John Shawe-Taylor. *An introduction to support vector machines and other kernel-based learning methods*. Cambridge university press, 2000.

[9] William Karush. Minima of functions of several variables with inequalities as side constraints. *M. Sc. Dissertation. Dept. of Mathematics, Univ. of Chicago*, 1939.

[10] Nasser M Nasrabadi. Pattern recognition and machine learning. *Journal of electronic imaging*, 16(4):049901, 2007.

[11] Thomas M Cover. Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition. *IEEE transactions on electronic computers*, (3):326–334, 1965.

[12] S. Sathiya Keerthi, Shirish Krishnaj Shevade, Chiranjib Bhattacharyya, and Karuturi Radha Krishna Murthy. Improvements to Platt's SMO algorithm for SVM classifier design. *Neural computation*, 13(3):637–649, 2001.

[13] Rong-En Fan, Pai-Hsuen Chen, and Chih-Jen Lin. Working set selection using second order information for training support vector machines. *Journal of machine learning research*, 6(Dec):1889–1918, 2005.

[14] Pai-Hsuen Chen, Rong-En Fan, and Chih-Jen Lin. A study on SMO-type decomposition methods for support vector machines. *IEEE transactions on neural networks*, 17(4):893–908, 2006.

[15] Neeraj Arora, Greg M Allenby, and James L Ginter. A hierarchical Bayes model of primary and secondary demand. *Marketing Science*, 17(1):29–44, 1998.

[16] Cho-Jui Hsieh, Kai-Wei Chang, Chih-Jen Lin, S Sathiya Keerthi, and Sellamanickam Sundararajan. A dual coordinate descent method for large-scale linear SVM. In *Proceedings of the 25th international conference on Machine learning*, pages 408–415. ACM, 2008.

[17] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: a library for support vector machines. *ACM transactions on intelligent systems and technology (TIST)*, 2(3):27, 2011.

[18] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830, 2011.

[19] Harvey Goldstein. Multilevel modelling of survey data. *Journal of the Royal Statistical Society. Series D (The Statistician)*, 40(2):235–244, 1991.

[20] Desmond L Nuttall, Harvey Goldstein, Robert Prosser, and Jon Rasbash. Differential school effectiveness. *International Journal of Educational Research*, 13(7):769–776, 1989.